INTERNATIONAL SPACE STATION TRAFFIC MODELING AND SIMULATION

THESIS

Jillene B. Rylaarsdam,
Second Lieutenant, USAF

AFIT/GOA/ENS/96M-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOA/ENS/96M-08

INTERNATIONAL SPACE STATION TRAFFIC MODELING AND SIMULATION

THESIS

Jillene B. Rylaarsdam,
Second Lieutenant, USAF

AFIT/GOA/ENS/96M-08

Approved for public release; distribution unlimited

AFIT/GOA/ENS/96M-08

INTERNATIONAL SPACE STATION

TRAFFIC MODELING

AND SIMULATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Jillene B. Rylaarsdam, B.S.

Second Lieutenant, USAF

March 1996

Approved for public release; distribution unlimited

THESIS APPROVAL

NAME: Jillene B. Rylaarsdam, Second Lieutenant, USAF **CLASS:** GOA-96M

THESIS TITLE: International Space Station Traffic Modeling and Simulation

DEFENSE DATE: 23 February 1996

COMMITTEE:	**NAME/TITLE/DEPARTMENT**	**SIGNATURE**
Advisor	E. Price Smith, Major, USAF Assistant Professor Department of Operational Sciences	
Reader	Richard F. Deckro Professor Department of Operational Sciences	

Acknowledgments

I first want to thank my Lord and Savior, Jesus Christ for helping me complete this degree. I also thank my friends and family for their constant support. I am extremely grateful for the insights, motivation, guidance and assistance provided by my advisor, Major E. Price Smith. Thank you to my reader, Dr. Richard F. Deckro, for keeping me on track and looking at the big picture. Finally, this research project would not have been possible without the sponsorship and support of Karen, Clare, Neil, Bob, Bob, and Jessica from Johnson Space Center. Thank you all for your help and support!

Jillene B. Rylaarsdam

Table of Contents

Table of Figures

List of Tables

Abstract

In an effort to provide NASA with an alternative perspective and some insights to the operational planning of the International Space Station (ISS), this research developed a simulation environment for the ISS and devised a method to evaluate various altitude strategies. The simulation environment allowed us to incorporate the natural random behaviors which affect the lifetime of objects in low-earth orbit. We created prototype models of the operational planning process to analyze current altitude strategy approaches and acquire new strategies from insights observed. In addition, by extrapolating random future solar activity values from the interpolation of historical data, we established a spectrum of possible solar activity rather than just maximum, mean, and minimum values. From this process, we demonstrated a procedure to analyze a strategy using distributions of parameter outputs in response to random inputs.

INTERNATIONAL SPACE STATION
TRAFFIC MODELING AND SIMULATION

1. Introduction, Motivation, and Background

1.1 PROGRAM OVERVIEW

The purpose of this thesis effort was to provide NASA with an alternative perspective and some insights to the operational planning of the International Space Station (ISS). At the request of NASA/OC, we looked at key issues involved in orbit and traffic planning and then created prototype models to represent operational issues. Our main focus in these prototype models was to directly account for random variations firmly embedded in the actual operational situation. In addition to a literature review in regard to concepts which form the backbone of this research, we also provided a brief literature review describing different OR methodologies to inform the reader of possible future areas of research that may be worth their time and finances to pursue.

1.2 INTRODUCTION TO THE SPACE ENVIRONMENT

On a clear night, void of lights from earth, we see the heavens above sparkle with lights that have intrigued humans for many years. Indiscriminate to race, gender, and even age, these lights beckon, tempting with their awesome complexity and the vastness

of the environment in which they exist. We gave the name of *space* to this enormous expanse beyond earth. Space. . . but what exactly is it? Many people have misconceptions of space. Some think that space only consists of the deep space, that is *way out there*. Others picture astronauts and objects like food and pencils floating about and believe there is no gravity in space. Still others view space as a huge vacuum, empty of all matter except for planets and stars. Finally, a few believe that once an object is launched into space, it will remain there for eternity.

Through the decades, scientists have dispelled these misconceptions. First, the space environment actually is closer than most people think. While no clear definition exists as to where space begins, a common starting point for space occurs only about 130 kilometers (80 miles) up from the surface of the earth. At this altitude, an object can briefly orbit the earth (Sellers, 1994).

Objects in space are not in zero gravity. All objects have gravitational forces which attract other objects. The force of the attraction depends upon the mass of the objects and their distance from each other. In fact, gravity provides part of the force that holds objects in orbit around the earth. Without gravity, they would continue in a straight line away from the earth. An orbit is really a balance between horizontal energy and gravitational pull. The energy of an object is a sum of its potential energy and kinetic energy. While potential energy is a function of an objections position, kinetic energy is a function of an object's mass and velocity. As an object increases in altitude, the

potential energy of an object increases. Kinetic energy increases as an objects mass and velocity increase.

The perception of zero gravity is achieved by the free-fall environment caused by the object *falling* around the earth. The object falls towards earth but, because of its energy, it continually misses the earth. This free-fall condition actually means that localized objects all have the same gravitational pull on them and are all *falling* at the same rate.

Space is not the complete vacuum many people identify. As one moves away from earth, the number of particles in a certain volume of space decreases until a near-vacuum environment exists (Sellers, 1994: 69). The amount of particles per volume of space is known as density. Although the condition of the atmosphere is *near-vacuum*, over time the small amount of density will slow the high velocity of the space station (Wiesel, 1989: 83). These atmospheric conditions of the atmosphere prevent objects in space from orbiting forever by reducing their potential enough for gravity to pull them to earth.

1.3 DRAG & EFFECTS OF DRAG

Objects slowly return to earth because drag causes a change in energy. We mentioned earlier that as distance from the earth increases, atmospheric density decreases. Since atmospheric density is one factor of drag, altitude also affects how much drag the object is subject to. Other factors include the mass, presented area (orientation and shape), and the speed at which the object is traveling (Sellers, 1994: 67). Atmospheric

drag opposes velocity and first affects an object by circularizing the orbit. As an object's altitude decreases, the velocity increases, thus increasing the kinetic energy of the object. However, the rate kinetic energy increases is smaller than the rate the potential energy decreases, causing a decrease in energy and, therefore, altitude (Sellers, 1994:112, 24:84). *Reentry* is the time in an object's mission where it is heating up quickly and impact is imminent because of a rapidly decaying orbit. Some parts will burn up, while others may even crash to earth.

1.4 Solar Activity & Affects on the Atmosphere

In addition to the effects on atmospheric densities due to a particular altitude, density varies based upon the amount of solar activity. Even though the sun may seem far away to some of us, it is our closest star and has a large effect on earth-orbiting objects. Besides the common elements of light and heat, the sun emits streams of charged particles, called solar wind, as well as solar flares (Sellers, 1994: 64). Solar flares are occasional huge bursts of charged particles. Sunspots, which are another mark of solar activity, are areas of extremely high magnetic fields. The amount of solar activity from the sun is constantly measured by scientists on earth. One common way to measure solar activity is by counting the number of sunspots, which relates to the frequency of disturbances (Tascione, 1988: 20). Although a high correlation exists between the number of sunspots and the frequency of disturbances, the complexity and indirect interactions between the sun and the near-earth space environment make predicted effects difficult (Gourney, 1990: 315). However, we do know that as solar

activity increases, the atmospheric temperature increases, and the layers of the atmosphere expand (Tascione, 1988: 61). The enlarged atmospheric span during high solar activity causes the density at a given altitude to be greater than at low solar activity. Solar maximum occurs when the average sunspot number is at its highest while solar minimum occurs when the average number of sunspots is at its lowest (Tascione, 1988: 19). Solar maximums and minimums occur on a cyclical basis ranging from 7-14 years. The strength of a solar cycle peak varies, as well as the length of a cycle.

During high levels of solar activity, the atmosphere in which an object is orbiting will become more dense than the nominal atmospheric density during low solar activity. This higher level of density will cause an object's orbit to decay faster. In fact, an object which would have remained in orbit for several years during average solar activity could be forced to reenter during a solar maximum. This unfortunate event actually happened to the first United States space station in 1973 (Project Skylab, 1992: Operations Summary).

1.5 Skylab Space Station

On May 14, 1973, the first United States space station, Skylab, was launched into low-earth orbit (Project Skylab, 1992: Operations Summary). Skylab used a converted third-stage Saturn V rocket from the Apollo Program and was built to demonstrate that humans could live and work in space for long periods of time and to extend our knowledge past earth-based observations (Project Skylab, 1992: Skylab Goals). After outliving it's intended lifetime, Skylab fell back into the earth's lower atmosphere on July

11, 1979, scattering debris over the Indian Ocean and parts of Western Australia (Project Skylab, 1992: Skylab Goals).

During the last mission to Skylab in 1974, the crew boosted the station high enough to remain in orbit for about nine more years (Oberg, 1992: 74). They also left some supplies in the unlikely case someone may return. Unfortunately, unexpected, intensive solar activity occurred, causing the earth's atmosphere to expand and Skylab to be pulled back to earth in 1979, much earlier than estimated (Oberg, 1992: 74). Although Skylab was never designed to be reused, the early reentry of Skylab motivates decision makers at NASA to plan for worst-case solar activity to prevent an unplanned reentry of the new space station.

1.6 Space Shuttle History

In 1981 the first space shuttle, Columbia, was launched. While the space shuttle program was the primary focus for NASA during the early 1980's, plans for a new space station were also under consideration. In 1984 President Ronald Regan announced that the Space Station Freedom (SSF) was to be built for various functional missions, including microgravity, life in space, and satellite repair (Easterbrook, 1991: 19). However, the space station was dependent upon the shuttle to transport the various parts for construction. Delays and technical difficulties plagued the space shuttle during the early years, followed by tragedy. In 1986, only five years after the first shuttle flight, the space shuttle Challenger exploded. Shortly thereafter, the U.S. lost a Titan 34D expendable rocket, a Delta first stage engine, and an Atlas Centaur launch vehicle

(Kolcum, 1986: 20). While the world watched with a jaundiced eye, NASA focused attention and funding on investigating and reviving the shuttle program. Due to political reasons as well as the fact that the shuttle is our only resupply vehicle, SSF never transitioned from theory to actuality. Ironically, also during 1986, the Russian Space Station Mir was launched. Since then, the Mir has been almost continuously manned, making the Russians the leaders in continuous manned space flight (McKenna, 1995: 18).

1.7 Space Station Benefits

According to NASA's Space Station White Papers, a space station is an orbiting research center created to study, explore, and better understand the environment of space, and America needs the space station for a variety of reasons (NASA White Papers, 1995: Part 1). These reasons include a research institute for cutting edge science, a space laboratory to provide new insights into life sciences, a catalyst for international peace and cooperation, a testbed for developing 21st century technologies, a springboard for global systems management, and a brighter future for our children and future generations, among others (NASA White Papers, 1995: Part 1).

A space station would help researchers solve scientific and technological problems that are of concern to us on earth (NASA White Papers, 1995: Part 1). Space stations exist in a low-gravity environment called microgravity. Under this environment, experimental measurements are more precise, mixtures are more uniform, and processes can be more clearly understood. For example, microgravity allows the development of high-quality protein crystals, which can be used to enrich our quality of life. Higher

quality drugs maximize effectiveness and minimize side effect, while purer semiconducting materials improve technologies like digital cellular phones, fiber optics, and high speed transistors and processors (NASA White Papers, 1995: Part 1). All these quality improvements are possible in microgravity environments.

The microgravity environment also gives new insights to life sciences. Biomedical research in space can help scientists understand different bodily functions as well as how diseases are spread. Some of the conditions include balance disorders, osteoporosis, and cardiovascular disease. With a better understanding of these and other diseases, scientists have a better chance of discovering cures or preventions.

Additionally, the space station is an ideal place to study the effects of long-term habitation in space. Currently, astronauts depend upon fresh supplies of water, food, and air from earth. In order to send crews out on longer missions, self-sufficiency will be required. Waste recycling is an important aspect for lengthy trips. The space station can provide an ideal climate for testing these concepts.

Another motive for the space station is the desire to remain on the front line of new global systems and technologies. Space station scientific and technological discoveries may well transfer over to other system integrations and spark more new developments as well as motivation for the next generation to continue the space exploration vision (NASA White Papers, 1995: Part 1). The answer to the space stations recycling problem could possibly provide valuable direction to solving environmental problems on earth.

Finally, the space station can foster warmer relations between countries in the Post-Cold War period. Currently, there are plans for a new space station to be built, called the International Space Station (ISS). Two of the proposed ISS missions will transfer over from the Space Station Freedom, including experiments in microgravity environments and the effects of space on the human body. The ISS is unique because it will be internationally planned, manufactured, launched, assembled, used and maintained. The United States and Russia are the largest players in the ISS, but 11 other nations plan to contribute to the station as well (NASA White Papers, 1995: Part 1).

1.8 International Space Cooperation

Although President John F. Kennedy suggested exploring the stars with the Soviet Union and other nations, the early 1960's were characterized by the competitive race to the moon between the United States and the former Soviet Union (Eastman, 1961: 46). The United States won this race on July 20, 1969, during the Apollo 11 mission when Neil Armstrong took "one small step for man, one giant leap for mankind."

Only six years later, during the frigid relations of the Cold War, the United States and the Soviet Union made history when the U.S. Apollo linked with the Soviet Soyuz in July, 1975. This event marked the first time these two countries united in space. Since this remarkable event of joint experiments took place during the height of the Cold War, hopefully a precedent has been set so that even if relations between participating nations grow tense, the ISS will remain an area of congeniality. With the end of the Cold War, warmer relations have emerged.

Campaigning to help keep the Russian space program alive and to bolster international cooperation in space, the United States will pay approximately $600 million dollars to Russia to use the Mir space station to observe various activities to ease the way for the joint ISS operations and for the purchase of a new space station module to be used on the ISS (Cowen, 1995: 312-313). Another reason to pursue ISS cooperation in financial support is to deter Russia from selling their knowledge of space exploration and missile technology to other countries to support their scientific infrastructures(Cowen, 1995: 312).

During the summer of 1995, the United States astronaut Norm Thagard commenced new relations when he joined the Russians in a Soyuz mission and subsequently docked with and spent some time on the Russian space station Mir. Thagard was the first American invited on a Russian mission. He actually lived and worked with the Russians until the shuttle Atlantis arrived on June 29, 1995 (Banke, 1995: 23). The Atlantis-Mir docking, only the second in history, marked the first union between the two primary space powers since the Apollo-Soyuz link in 1975. In addition, Atlantis carried two Russian cosmonauts aloft to replace the ones aboard the Mir with Thagard. The replaced Russian crew returned to earth aboard the Atlantis and to the United States - another first in international space flight. This connection was only the beginning of a series of linkups designed to help prepare for the construction of an international space station, scheduled to begin in 1997 (Cowen, 1995: 312).

1.9 ISS & Dependence Upon Russia

The ISS program is extremely dependent upon continued Russian involvement. In fact, the ISS will consist of parts that were originally designated for the second-generation Mir (Cowen, 1995: 312, ; Cowen, 1993: 399). The fabrication of the ISS is slated to take five years from the first launch. While the manufacturing of parts will be done on earth, the space station will be launched in segments and assembled in orbit. Over forty flights are required to complete the space station.

The first ISS segment to be launched is the functional energy block (FGB), which was Russian made and tested, but is now owned by the United States. This self-sufficient orbital vehicle will be launched by Russia and will provide the primary fuel storage capability as well as initially perform the station reboost and attitude control (ISSA Reference Guide, 1995: 73,171). The FGB has a capacity to hold 6120 kg of propellant. The second Russian flight will carry the Russian Service Module (SM). Once the SM is in place, the FGB will only be used for the storage and transfer of propellant (ISSA Reference Guide, 1995: 73).

The SM has 860 kg of fuel storage capacity and contains the dock where the Russian vehicles will dock. It will take over the duties of attitude control and reboost from the FGB until a Russian vehicle Progress M or Progress M2 is attached, at which point the SM will be used as backup to the Progress (ISSA Reference Guide, 1995: 73, 182). Reboosts will then be performed by Russian Progress vehicles docked to the SM. The Progress is currently the only vehicle used to transfer propellant to the space station.

Progress can transfer fuel to either the SM or the FGB. Because of their respective locations, reboosts using the Progress will conserve more fuel than reboosts executed with the SM (ISSA Reference Guide, 1995: 73). The present design states that when the attached Progress is empty, the vehicle will de-orbit and a new Progress will be launched (ISSA Reference Guide, 1995: 73). According to NASA contractors, another possible design allows the FGB and SM to transfer propellant to the empty Progress for more efficient usage. Since the space station only receives fuel from Russia, problems may arise in maintaining fuel levels should this international situation adversely change or if problems arise in the Russian's space program. Currently, the U.S. does not have the capability to deliver fuel to the ISS, but will bring the U.S. crew, crew supplies, logistics, experiments, and other cargo.

1.10 Funding & Political Issues

In addition to the international politics inherent in the design and building of the ISS, domestic U.S. politics strongly influence funding for the space station. Funding is a high concern for NASA because Congress is becoming restless with the project. This is the seventh proposed design in over ten years (Cowen, 1995: 312). Currently, the scheduled time to begin construction falls just as the solar cycle is predicted to begin another drastic rise and ends at the estimated solar maximum. In practical terms, this means that the space station will need more reboosts to prevent reentry than if the construction period was at a solar minimum. This timing is not desirable because the main focus of the station during this period is launching segments for construction, not

worrying about trying to keep the station from falling back to the earth. Any delays will push the program even further into the height of the predicted cycle. A better time to begin construction would be about the year 2004, just as the solar activity calms down and approaches a solar minimum. However, this long of a schedule slip is unacceptable from a mission perspective. In addition, a general perception at NASA is that the funding may not be continued if the space station does not go as scheduled. Finally, while the solar activity may be undesirable for construction, benefits include lower levels of solar activity after construction and thus longer periods of micro-gravity for experimental purposes.

1.11 Operational Plan and Planning Teams

Looking past the issues of funding, steps need to be taken under consideration to make certain that the ISS does not meet the same fate as its predecessor, Skylab. In order to anticipate this problem, operational plans must be constructed to ensure that the space station's mission is accomplished over its intended lifetime. Figure 1.1 illustrates the basic flow that occurs during this process.

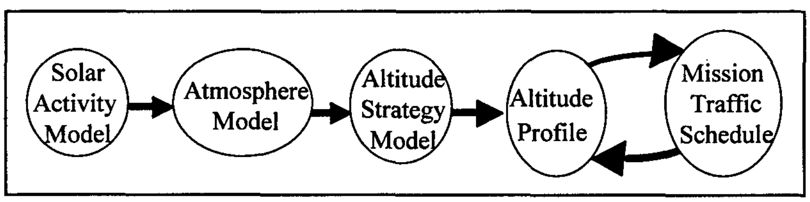


FIGURE 1.1 Basic Flow Chart of Operations Plan

First, solar activity is monitored to give a first impression on the type of altitude and reboost strategy that will probably be needed. For example, if we observe that the

solar activity is going to be high for the next several years, we will know that higher altitudes will be required and/or more reboost events scheduled to keep the ISS from reentering into the atmosphere and crashing into earth. Second, the altitude and frequency of reboosts will give indication of the amount of propellant needed. Finally, the amount of propellant will govern the number of fuel vehicles needed which could, in turn, regulate the vehicle traffic schedule. The traffic schedule also includes docking limitations and microgravity requirements. This phased loop attempts to find a feasible plan which will keep the space station from disaster.

Within NASA, there is a traffic modeling team whose responsibility is to create the operations plan. Besides coordinating the amount of propellant to sustain altitudes, this plan must also plan for the docking of all international flights while considering periods of microgravity. Long term concerns of NASA are to make certain this station stays in orbit and meets its requirements for microgravity and crew supply. The ISS traffic modeling team is composed of two different sections: a design analysis team and an operations planning team.

1.11.1 Design Analysis Team

The first section of the ISS traffic modeling team is a design analysis group. The primary concern of this group was to determine the feasibility of the space station by creating an initial schedule for the estimated fifteen year life span of the space station. They created an operations plan that included the different constraints involved in the problem, including altitude limitations inherent in station design. The operational plan

includes an altitude strategy, altitude profile, reboost strategy, propellant requirements, and a traffic schedule. This strategy plans for the worst-case solar cycle, but counts on a predictable reboost cycle. Only one missed reboost, called a *skip-cycle*, is allowed in their planning. NASA requires the space station to have sufficient skip-cycle propellant to reboost to an altitude that results in at least one year of orbital decay to 278 km under nominal operations (Puckett, 1994:9). This skip-cycle capacity may either be used for one large reboost or several smaller reboosts.

The space station will orbit at an altitude approximately 300-460 kilometers. At these altitudes, one significant factor, as well as the most uncertain, of the space environment includes the density of the atmosphere (ISSA Reference Guide, 1995: 98). An altitude strategy is a set of guidelines and assumptions needed to decide where the space station will operate (McDonald, 1990: 2). The rate the space station will decay back into the atmosphere is proportional to the atmospheric density (McDonald, 1990: 2).

Space stations operate in an altitude range defined by design limitations on the upper end of the altitude and a safety zone to prevent reentry on the lower end of the altitude. While the minimum allowable altitude varies based upon solar activity, the maximum allowable altitude is constant, based upon Russian design constraints. As solar activity increases, the earth's atmosphere expands and becomes more dense and the minimum altitude increases. The maximum altitude authorized by the Russian Space Agency for hardware limitations is 460 kilometers (Loyd, 1995: 6). Depending on the solar activity, the range between the minimum safety and the maximum hardware

limitations can be narrow. For example, the predicted solar activity around September of 2000 will set the minimum altitude at about 410 kilometers. In order to the keep the space station in its allowable altitude range, reboosts must either occur more frequently or to higher altitudes. The rule of thumb altitude for the Russian Soyuz to rendezvous is 425 kilometers. This altitude ensures that the Soyuz has enough propellant to deorbit, as propellant cannot transfer to the Soyuz.

Conversely, as solar activity decreases, the atmosphere contracts and becomes less dense and the space station can remain in orbit for longer periods at lower altitudes. The altitude planner has several objectives to consider in planning an altitude strategy. Some of these objectives include planning simplicity, disturbance levels, minimizing rendezvous altitudes, optimal altitudes, constant propellant and shuttle decay (McDonald, 1990: 3-4, Koepke, 1992: 2-3). From a given strategy, altitude profiles can be calculated.

The altitude profile will designate lower reboost altitudes, rendezvous altitudes, and skip-cycle calculations. These figures will then determine an estimation of the amount of fuel needed, which will dictate how many Progress M or Progress M2 vehicles are needed to reboost the station. The Progress vehicles will then figure into a schedule, along with other variables such as upmass, required cargo, microgravity, vehicle loading, and docking limitations with traffic from other countries. A schedule is then determined, which feeds back into the altitude strategy. The calculations for the different stages of this planning loop are done using multiple computer models. These computer models were created by the long-term planning team to aid them in developing faster results to be

used in decision making. User interpretation is vital. Iterations require manual interactions.

The first model in this loop takes an altitude strategy and generates altitude profiles. This model is a Station Reboost Analysis Program, or STRAP. STRAP is written in the C programming language and runs in a UNIX environment. STRAP gives the user six different strategies to specify the lower altitudes for reboost and skip-cycle calculations. After reading in the altitude strategy(ies), STRAP calculates reboost altitudes, rendezvous altitudes and propellant estimates needed for the reboosts (Delaney, 11/13/1992:1, Koepke, 1992:1). The required propellant will determine how many progress flights are needed. Finally, the required flights will be one of the many inputs to the Traffic Model.

The current version of the Traffic Model (TM) runs on an Excel spread sheet and requires a great deal of interpretation by its developer (Lemmons, 1995). The Traffic Model identifies vehicles visiting the ISS, upmass, microgravity, and required cargo. It then calculates detailed schedules and projects a guideline for resupply. The model also classifies who has responsibility for deliveries and computes a profile for when the station is not to be disturbed. Figure 1.2 (below) attempts to graphically capture the flow of the operations plan.

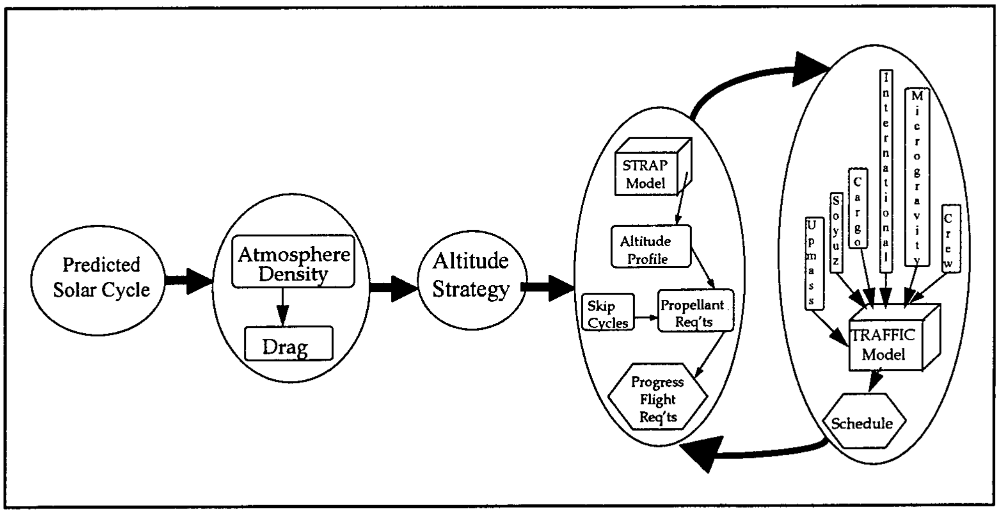


FIGURE 1.2 Detailed Flow Chart of Operations Plan

1.11.2 Operations Planning Team

The second section of NASA/OC traffic modeling team plans ISS operations for long-term scheduling. This schedule utilizes a horizon of approximately five years with a plan that responds to events that happen year by year. NASA/OC now is in the process of transitioning the models from design analysis tools to operations analysis tools. Now that the concept of the space station and different altitudes has been proven feasible, the operations planners will modify these tools to implement actual decisions for the station. While the design analysis planners used these tools to gain a static snapshot, the mission of the operations planners is different. Knowing beforehand that the intrinsic randomness found in the environment of the space station and the launch schedule will change the design plan, the operations planners need to consider this randomness and incorporate it into the models. During this transition stage, NASA is taking a step back to determine how they can look at the situation from a different perspective. By looking at the

problem from an Operations Research perspective, NASA hopes to gain different insights to help in decision making.

1.12 Operations Research

The field of Operations Research (OR) crystalized during World War II in order to assist warplanners in assigning limited resources in the most effective manner (Hillier, 1990: 4). Since then, the field has greatly expanded and now can be divided into several areas. Although they can overlap, deterministic and probabilistic methods are two main classifications of OR approaches. A primary deterministic OR method is mathematical optimization, which seeks to assign the controllable aspects of a system (variables) in order to maximize or minimize a specific objective such as profit or cost, subject to various constraints. Optimization methods are useful for solving problems such as scheduling, allocation, assignment, and transportation. The potential exists to use optimization for the scheduling and vehicle packing problems currently solved using computational, logical spreadsheets and heuristic human-in-the-loop processes. Parameters are assumed to be fixed and do not vary, except in post optimality sensitivity and parametric analysis. Randomness usually does not appear in these types of models.

Probabilistic methods, on the other hand, incorporate randomness into the analysis to get an idea of how outputs can change by varying inputs or giving the inputs a probabilistic nature. Probabilistic methods are useful for solving problems such as queuing, forecasting, decision analysis, inventory analysis, and steady-state analysis of random systems. Simulation is a tool which allows the user to perform detailed analyses

of systems with complex interactions of random components. Computer-based simulations are commonly used techniques when modeling randomness is too difficult, or impossible, to do analytically or when many detailed computations are needed to accurately depict the effects of random system interactions.

While NASA is in this stage of transferring the long-term models to short-term models, our research aims to give them a different perspective of the problem. Until now, they have been looking at the problem from an engineering design point of view, including worst-case analysis, long range feasibility studies, models of equations of motion and fixed-environment assumptions. This view uses certain laws of nature like scientific formulas and assumes fixed values. We know that the outputs of these models are hypothetical and, since they address the worst-case scenarios, are not likely to occur. Building for the worst case scenario is a design imperative, but an operational planning roadblock. We wish to evaluate orbital and scheduling strategies by investigating the possible outcomes that the randomness in the system might cause. The Operations Research modeling process can take into account the randomness in the system such as solar activity, atmosphere density, drag, vehicle slips, failures and/or cancellations. Modeling randomness has the potential to improve efficiencies of operations by providing a method for evaluating various operational planning strategies.

The purpose of our research is to help NASA look at their problem from a different perspective and help them identify possible approaches. We will use probabilistic methods of Operations Research to model the randomness inherent in ISS operations to demonstrate improvements in short-term orbital, traffic, and vehicle

planning. By modeling these different aspects, we hope to provide NASA with an appreciation of how incorporating OR methodologies into their atmospheric and traffic modeling can help improve their operational planning process.

2. *Literature Review*

2.1 SCOPE

The primary purpose of this research is to provide NASA with an alternative approach in considering their analysis of the interactions between components affecting the International Space Station (ISS). Some of these components include the natural random levels of solar activity, vehicle slips, and decisions of when to reboosting the station. To do this, we first need to understand the basics of the models NASA uses to design an operational plan for the ISS and simplify them to a manageable size. Our proposed Iteration Approach creates prototype models that represent the iterative flow of data through the operations planning process. These models will illustrate related concepts on simplified, aggregate examples.

In this review we will explore the literature relating to different aspects of modeling the life of the ISS and will discuss the applicability of the approach to our research. The different aspects include solar activity, atmospheric conditions, altitude strategies, altitude profiles, propellant requirements, and traffic models. In addition, simulation will be discussed, as this is our primary research tool for this study. Chapter 5 will discuss other Operations Research approaches that may be considered in future research on these issues.

This study is relevant to NASA because it will suggest new ways of viewing their planning process by directly planning for randomness. By planning for randomness, they can reduce the large buffers of error and increase upmass and fuel consumption.

2.2 Solar Activity

As the earth's closest star, the Sun has the largest effect on objects in the earth's atmosphere. Most of the sun's energy is in the form of low-energy photons and, compared to higher frequency energy, has a fairly constant output (Withbroe: 394). However, the energy output in the higher end of the spectrum causes solar activity recordings to vary widely (Withbroe: 394). Huge, short, intense bursts of charged particles, called solar particle events or solar flares, also sometimes occur. Sunspots, another form of solar activity, are areas with strong magnetic fields. Sunspots are often used in the recording of solar activity because they were easy to observe are proven to be a *good* index of activity (Withbroe: 394). New technology for measuring solar activity gives results which are highly correlated to the sunspot number (Withbroe: 394).

The Zurich index is one of the most common indicators of solar activity. This index measures the number of sunspots and sunspot groups observed on the sun and takes into account a correction factor for observation error. This solar activity has a strong affect on the density of the atmosphere. Since the atmospheric density affects the time an object will remain in orbit, prior knowledge of the amount of solar activity is extremely useful. However, because of the unpredictability of solar activity and the complex interaction with its effects on the earth's lower atmosphere, accurate predictions of solar

activity and its effects are virtually impossible. Attempts to capture the fundamental aspects of solar activity continues.

Schwabe (1843), as translated by Meadows, reviews his finding from the years 1826-1843 and discusses the possibility that the number of sunspots may have a period of about 10 years. This was the first discovery of a solar cycle.

Tascione (1988) discusses that more recent conclusions have found the solar cycle has an average length of 11 years, but can vary from 7 to 13 years. In addition, he comments that the a representative cycle will rise from solar minimum to solar maximum in 4 years and then fall back to solar minimum in 7 years. Tascione then introduces the Zurich sunspot number, *R*. This number is commonly used in recording solar activity and reflects the number of spots and the number of sunspot groups on the sun.

Withbroe (1988) also describes the sunspot cycle, but states that the 10.0 years is the shortest the cycle has been since 1850 while 12.1 years is the longest (Withbroe: 394). There also was a period between 1645 to 1715 when solar activity about disappeared (Tascione: 19). This duration of inactivity, known as a *Maunder minimum* may occur on an irregular basis (Tascione: 19). Withbroe furthermore determines that the average time from minimum to solar maximum is 4.3 years and it takes 6.6 years to fall back to solar minimum (Withbroe: 394). After reviewing other articles, Withbroe concludes that the faster a cycle rises to solar maximum, the higher its maximum value. The author mentions that smoothed data across 12 to 13 months are commonly used in

place of the highly fluctuating daily sunspot number. Thompson (1995) mentions that the sunspot numbers chart smoothly when averaged.

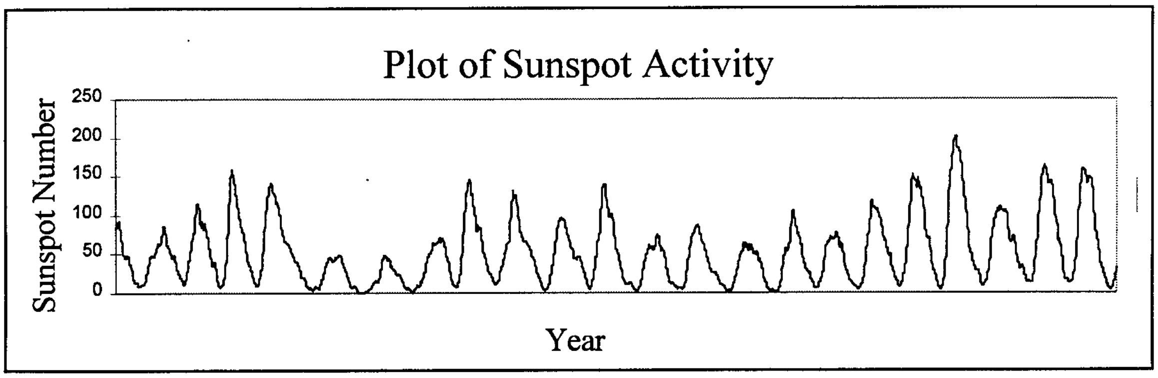


Figure 2.1 Representative Sunspot Cycle Graph (Thompson, 1995)

However, he also states that there are variations in the yearly curve due to rapid regions of solar growth associated with geomagnetic activity such as solar wind and solar flares.

According to Wiesel (1995) solar flares can increase the measurements of solar activity by a factor of 10. Gorney (1990) refers to the increased flux of energetic particles as solar particle events (SPEs). He comments that although only a few of these events occur per year, they have a large effect on the electronics on satellites and can cause illness in astronauts. These SPEs can last anywhere from a few hours to days. While SPEs can occur any time during the solar cycle, other than at solar minimum, Gorney observes that the frequency of SPEs seems to peak from 2 years prior to 4 years after sunspot maximum.

As modern technology increased, better means for observing solar activity has resulted. Withbroe introduces the solar 10.7-cm radio flux as a measurement of how bright the sun is when observed at a wavelength of 10.7 cm (Withbroe: 394). This

method of measuring solar activity has been recorded since 1947. The author gives the relationship between the radio flux and 13-month smoothed means of the Zurich sunspot number R as the following:

$$R = 1.075 \cdot F_{10.7} - 61.1 \qquad \textbf{(2.1)}$$

NASA (TM-82478) uses a different equation than Withbroe in the conversion of smoothed sunspot data (R_Z) to smoothed solar flux data ($\overline{F}_{10.7}$):

$$\overline{F}_{10.7} = 49.4 + 0.97 \cdot R_Z + 17.6 \cdot \exp(-0.035 \cdot R_Z) \qquad \textbf{(2.2)}$$

Again, the amount of solar activity relates to atmospheric conditions and directly affects how long low-earth-orbit objects remain in orbit. If not accounted for in modeling effects, large errors in orbital analysis will likely occur and objects may not remain in orbit as long as planned. Hence, solar modeling is a critical component of any low-earth-orbiting object.

Richard Thompson for IPS Radio & Space Services in Sydney, Australia provided historical data via electronic mail on the Internet from July 1749 through August 1994. Predicted values were also given through December of 1997. The values given are known as *Smoothed Monthly Mean Sunspot Numbers*. In his note, Thompson defines the smoothed number as "the average (division by 12) of thirteen values with the 1st and 13th values being given half weight." He gives an example of the June 1980 value which was created by taking the monthly sunspot number values from December 1979 through December 1980. Each of the December values were given half the weight and then added

to the sum of the other eleven numbers, January 1980 through November 1980. The entire sum was then divided by twelve.

2.3 Atmosphere Models

2.3.1 Density Models

While solar activity is a large contributor to atmospheric density, other factors will affect it as well. Tascione discloses that, in addition to solar flux and geomagnetic activity, atmospheric density is affected by local time, altitude, and latitude. He further concludes that an average of the density over local time and latitude will expose a semiannual period with the largest height in October and a second peak, although not quite as large, in April. NASA gathers that density varies between the seasons and the peak of density in the winter is a result of higher concentrations of helium (TM-82478, 2-13). Due to the many factors affecting atmospheric density, many of which are virtually impossible to predict, an accurate model of atmospheric density is unattainable. Withbroe states that there are currently no accurate long-range solar activity prediction methods (Withbroe: 399). All models use simplifying assumptions in order to construct representative values for atmospheric density.

One of the more complicated models of atmospheric density is the Jacchia-Lineberry Upper Atmospheric Density Model (1982). The J/L model incorporates many of the important characteristics of the upper atmosphere and is assumed valid over the altitude range of 90-2500 km. The main drawback of this model is the large

computational expense associated with generating densities. This model closely relates to a previous model, the Jacchia 71 Model. Results are compared to another earlier model, the Jacchia 70 Model, as well.

According to McDonald and Teplitz of McDonnell Douglas (1990), the Jacchia 1970 atmospheric model was the accepted model for the Space Station Freedom Program. This model mainly used the solar flux ($F_{10.7}$) and the geomagnetic index (K_p) in the calculation of atmospheric density (McHenry: 3). The geomagnetic index is an indicator of "...the *general level* of magnetic activity caused by solar wind," (Tascione: 40).

Simpler models for calculating atmospheric density use solar activity and altitude as their two parameters. One reason could be that the geomagnetic activity relates closely to the sunspot number (Gorney, 321). Another view, held by Walterscheid (1989), is that geomagnetic disturbances do not significantly affect satellite lifetimes because of their brevity. Regardless, Walterscheid and NASA (TM-82478) plot atmospheric density as a function of altitude and solar activity and Hedin (1986) charts atmospheric density as the same. These models illustrate that atmospheric density is a factor of altitude and solar activity. The higher the levels of solar activity, the higher the density. Conversely, the higher the altitude of an object, the lower the density encountered. As our research does not contain enough detail to use the J/L model, we will attempt to model atmospheric density to resemble the graph of NASA and chart of the Larson and Wertz.

2.3.2 Drag Models

Density is the key parameter, as well as the most uncertain, in the calculation of atmospheric drag (McDonald/Teplitz: 2). The acceleration an object receives due to drag is also known as the decay of an object's altitude. A second parameter for calculating drag is the ballistic coefficient, which McDonald and Teplitz describe as how an object resists orbital decay. The ballistic coefficient is a function of the presented area, or cross-sectional area, the mass of the space station, and a coefficient of drag.

Walterscheid (1989) asserts that the ballistic coefficient is a function of the composition and temperature of the atmosphere. However, he also claims that while atmospheric density can cause variations in drag by one order of magnitude, other factors will not usually alter the drag by more than about 10%. This conclusion assumes that an orbiting object will assume a constant area-to-mass ratio.

In fact, all formulas in this review define the ballistic coefficient as some form of the following:

$$B = \frac{m}{C_d \cdot A} \tag{2.3}$$

where

m = Mass of the object (kg)

C_d = Drag coefficient

A = Presented area of the object (km^2)

When defined as Equation 3, the higher the ballistic coefficient of an object, the slower it tends to decay. According to McDonald and Teplitz, 2.3 is a typical drag coefficient for

an orbiting space station (McDonald/Teplitz: 3). In addition, under their standard conditions for the SSF, the ballistic coefficient equals 12 lb_f/ft^2, or 58.59 kg/m^2. The STRAP Programmer's Guide (1992) and User Handbook (1994) both use 12 lb_f/ft^2 as well. However, contractors from NASA report that the ISS has a ballistic coefficient of 14 to 15 lb_f/ft^2. For our research, we will convert 14 lb_f/ft^2 to 63.35 kg/m^2 for our models.

Wiesel instructs that drag will first circularize the orbit of an object. Therefore, we will assume circular orbits before calculating decay. Given this assumption, Wiesel provides the following equation motion:

$$\frac{da}{dt} = -\sqrt{\mu \cdot a} \cdot B^* \cdot \rho_o \cdot e^{-(a-Re)/h} \tag{2.4}$$

where

a = Distance from the center of the earth (km)

B^* = Ballistic coefficient (km^2/kg) ($B^* = 1/B$ of equation 3)

ρ_o = Fictitious base density of the atmosphere (kg/km^3)
(*Fictitious* ≡ User defined base density)

R_e = Radius of the earth (km)

h = Atmospheric scale height (km)

μ = Gravitational parameter (km^3/s^2)

Since Wiesel defines $\rho = \rho_o \cdot e^{-(a-R_e)/h}$ for a circular orbit, we will solve for ρ_o and substitute to obtain the following equation:

$$\frac{da}{dt} = -\sqrt{\mu \cdot a} \cdot B^* \cdot \rho \qquad (2.5)$$

Wiesel defines the gravitational parameter $\mu = G(m_1+m_2)$, where m_1 is the mass of the earth and m_2 is the mass of the orbiting object. Since the mass of the orbiting object is insignificant to the mass of the earth, the author reduces μ to Gm_1. He cites this value as $3.98601 \times 10^5\ km^3/s^2$. The radius of the earth is equal to 6378.135 km. This is the equation we used in our model.

Boden (1992), under the assumption of circular orbits, formulates changes in satellite altitude, orbit period, and satellite velocity per revolution:

$$\Delta a_{rev} = -2 \cdot \pi \cdot B^* \cdot \rho \cdot a^2 \qquad (2.6)$$

$$\Delta P_{rev} = -\frac{6 \cdot \pi^2 \cdot B^* \cdot \rho \cdot a^2}{V} \qquad (2.7)$$

$$\Delta V_{rev} = \pi \cdot B^* \cdot \rho \cdot a \cdot V \qquad (2.8)$$

$$\Delta e_{rev} = 0 \qquad (2.9)$$

where

Δa_{rev} = Change in satellite orbit per revolution (km)

ΔP_{rev} = Change is orbit period per revolution (s)

ΔV_{rev} = Change in satellite velocity per revolution (km/s)

Δe_{rev} = Change in orbit eccentricity per revolution

B* = Ballistic coefficient (km^2/kg) (1/B of equation 3)

ρ = Atmospheric density (kg/km^3)

a = Distance from center of earth to satellite altitude (km)

From Boden's equations, Larson and Wertz formulate mean and maximum decay rates in kilometers per year:

$$\textit{orbit decay rate (km/yr)} = \frac{-2 \cdot \pi \cdot \left(\frac{C_d \cdot A}{m}\right) \cdot \rho \cdot r^2}{P} \tag{2.10}$$

where

$\frac{C_d \cdot A}{m}$ = Ballistic coefficient (km^2/kg) (1/B from equation 3)

ρ = Atmospheric density (kg/km^3) (use either mean or maximum)

P = Period (min)

r = Distance from the center of the earth

The tables in the back of their book provide actual values for mean and maximum orbit decay rate, depending of which density value is used. Larson and Wertz express the period in minutes and convert it to years for the equation.

2.4 Altitude Strategies

Once the rate of decay is determined, plans are made to reboost an object to prevent it from reentry. McDonald and Puckett (1991) define an altitude strategy as a "set of (design, operational, and planning) guidelines and assumptions used to determine an altitude regime for SSF operations," (McDonald, 1991:8). McDonald and Teplitz

(1990) give four different altitude strategies which were considered for the Space Station Freedom. In 1992, Koepke and McDonald offer two additional strategies for use in the Station Reboost Analysis Program (STRAP). The four similar strategies, discussed in the following paragraphs, include: constant altitude, constant micro-gravity altitude, constant lifetime altitude, and optimal altitude. STRAP adds constant propellant and shuttle decay.

2.4.1 *Constant Altitude* Strategy

For this strategy, rendezvous altitudes are conducted when the station reaches a certain altitude (specified by the decision maker). The advantage of this strategy is simplicity. McDonald and Teplitz (1990:3) stress that the chosen altitude must not violate any requirements during the course of the solar cycle. This constraint forces the chosen altitude to be based upon *maximum* solar activity. As solar activity is at a maximum for only 6 to 18 months, the authors point out that the simplicity of this strategy gives up additional payload-to-orbit capability during times when the solar activity is lower. They also remark that while this strategy has constant payload-to-orbit, the allowable reboosts contain large variations, which complicate manifest planning. Besides the disadvantage of limited operational flexibility, McDonald and Teplitz state that additional flights are needed to use this strategy.

2.4.2 *Constant Micro-Gravitational Level* Strategy

For this altitude strategy, rendezvous altitudes are performed at a constant atmospheric micro-gravity level, or rate of decay. McDonald and Teplitz assert that the

advantage of this strategy is that it allows lower rendezvous altitudes during periods of decreased solar activity (4). Lower rendezvous further correspond to increased payload-to-orbit capabilities which also result in lower life cycle costs during the life of the station. Since the decay rate is a factor in both rendezvous and reboost altitude, another advantage of reboosting at a constant decay rate is that station propellant requirements for reboost are somewhat constant. The authors point out that a large disadvantage for this strategy is that the large variations in the solar cycle sometimes resulted in unsafe margins for orbit lifetimes.

2.4.3 *Constant Lifetime Altitude* Strategy

This altitude strategy performs rendezvous at the minimum allowable lifetime level. The advantage of this strategy is that shuttle payload-to-orbit capability is maximized. However, this strategy gives no margin for launch slips or increased atmospheric activity.

2.4.4 *Optimal Altitude* Strategy

For this altitude strategy, rendezvous occurs at an altitude where net payload-to-orbit is maximized. McDonald and Teplitz define net payload-to-orbit as the total shuttle delivery capacity minus SSF reboost propellant requirements (4). At a lower rendezvous altitude, the shuttle has an increased payload-to-orbit capability, but the SSF requires more propellant for reboost. Conversely, at a higher altitude, the shuttle has less payload-to-orbit capability, but the SSF requires less propellant for reboost. The authors define the optimal altitude as:

> . . . the altitude at which flying lower would cause more additional propellant to be used than gained in Space Shuttle payload-to-orbit, and flying higher would cause more Space Shuttle payload-to-orbit lost than would be saved in reduced propellant needs (McDonald, 1990:4).

McDonald and Teplitz assess that this strategy is insensitive to atmospheric predictions. In addition, the net payload-to-orbit decreases slowly as the actual altitude deviates from the optimal altitude.

2.4.5 *Constant Propellant* Strategy

This strategy uses a rendezvous altitude of the previous flight and the determines the magnitude of the reboost by the amount of propellant specified along with the specific impulse (I_{sp}) of the propellant.

2.4.6 *Shuttle Decay* Option

The Shuttle Decay is actually an option instead of a strategy. This option allows the decision maker to alter the ballistic number and vehicle configuration to reflect realistic events such as when the vehicles are docked verses when the station is orbiting solo. Such a change will affect the decay rate.

After reviewing and testing some of these altitude strategies, we propose an altitude strategy which is dependent on atmospheric predictions. This will allow a strategy that is responsive to the primary random factor in the orbital planning. Thus, we can perhaps improve the orbital planning characteristics, in particular the ability to remain within safety margins, microgravity window requirements and shuttle upmass performance (enhanced by docking at *low* altitudes).

2.5 ALTITUDE PROFILES

Once a decision maker has chosen the altitude strategy (or strategies), the next step is to generate altitude profiles. An altitude profile designates lower reboost altitudes, rendezvous altitudes, and skip-cycle calculations. From these specifications, an altitude profile determines when a reboost takes place and the length of the reboost. The reboost event estimates the amount of fuel needed, which establishes how many Progress M or Progress M2 vehicles are needed to boost the station.

2.5.1 STRAP Overview

The Station Reboost Analysis Program (STRAP) mentioned earlier is an analysis tool which creates altitude profiles based on given altitude strategies. Koepke and McDonald (1992) list several steps STRAP uses to generate an altitude profile. First it determines the lower altitudes devised from the first four altitude strategies (*altitude*, *micro-gravity*, *lifetime*, and *optimal*). Since the *shuttle decay* option requires a previous lower altitude from which to back up, step two is to determine the *shuttle decay* altitudes that follow altitudes calculated in step one. Next, STRAP computes the *constant propellant* lower altitudes. These altitudes are created third in the sequence because this strategy needs the previous lower and upper altitudes. Be aware that this strategy uses the previous non-*shuttle decay* lower altitude. STRAP then adds any remaining *shuttle decay* options that had to wait for the *constant propellant* altitudes. For the last step, STRAP calculates the remaining upper altitudes.

One aspect to notice is STRAP calculates reboosts on a constant interval basis. A typical interval is 90 days. From these intervals, STRAP creates the altitude profiles. Delaney (1994) states that upper altitude is created by starting with an initial guess and then using the Newton-Raphson iteration technique (7).

STRAP also allows the user to have the program calculate skip-cycle propellant requirements. Koepke and McDonald state that this option permits a flight to slip by an interval defined by the user. The user then has the choice of *lower altitude, micro-gravity level, lifetime,* and *optimal* strategies (see sections 2.4.1 through 2.4.4) for STRAP to use in computing the skip-cycle minimum altitude. From this altitude, STRAP will compare the upper altitudes from the skip-cycle and the original calculation and take the larger of the two. The previous reboost event will boost the station to that altitude. Unfortunately, this technique assumes that prior knowledge is available for a slipped mission. However, NASA does assume that the Space Station shall have enough on-orbit reserve propellant for a reboost to an altitude that results in a specified number of days (i.e. 90, 180, 360) of orbital decay to 278 km under nominal operations (Sanders). If the reserve propellant required exceeds the capacity of the station, the amount of on-board propellant will determine the boost. Delaney adds that the skip-cycle altitude may never violate the minimum altitude constraint.

2.5.2 Propellant Requirements

According to Delaney, STRAP uses the ideal rocket equation assuming a Hohmann transfer formulation to calculate the required reboost propellant. McDonald and Teplitz (1990) list this equation as the following (13):

$$Prop = Mass \cdot \left[1 - \exp\left[\frac{-\Delta V}{G_e \cdot I_{sp}}\right]\right] \tag{2.11}$$

where

$Mass$ = Total SSF mass (kg)

G_e = Acceleration of gravity at the earth's surface
= 9.8 m/s^2

ΔV = Velocity change required to achieve a circular target orbit based on the height of the reboost (m/s)

I_{sp} = Propellant specific impulse (sec)

Delaney specifies that the propellant specific impulse is assumed constant. McDonald and Teplitz list 230 seconds as the standard condition for I_{sp}. From the ISSA Reference Guide, the ISSA on-orbit weight after assembly is 924,000 lbs, or 419,119 kg.

From Boden (1992) we obtain the following equations for ΔV assuming a Hohmann Transfer:

$$\Delta V_{Total} \equiv \Delta V_A + \Delta V_B \tag{2.12}$$

$$\Delta V_{Total} = \sqrt{\mu} \cdot \left[\left|\left(\frac{2}{r_A} - \frac{1}{a_{tx}}\right)^{1/2} - \left(\frac{1}{r_A}\right)^{1/2}\right| + \left|\left(\frac{2}{r_B} - \frac{1}{a_{tx}}\right)^{1/2} - \left(\frac{1}{r_B}\right)^{1/2}\right|\right] \tag{2.13}$$

where

μ = Gravitational Parameter
= $3.98601 \times 10^5\ km^3/s^2$

r_A = Initial Orbit (km)

r_B = Final Orbit (km)

a_{tx} = Semi-major axis of the transfer ellipse (km)
= $(r_A + r_B)/2$

Puckett mentions that a 5% uncertainty calculation is then added to the ΔV to accommodate the difference between the assumed impulsive burn and a finite burn, engine I_{sp} uncertainties, potential weight growth, and thruster misalignment, to name a few possible sources of differences.

The required propellant cannot exceed the combined propellant of the on-board Space Station capacity and the available propellant from the Progress vehicle (if no Progress vehicle is attached, the available propellant is zero). Puckett (1995) provides the following information concerning the on-board propellant and Progress vehicle propellant (see Table 1).

TABLE 1

PROPULSION ELEMENT PERFORMANCE CHARACTERISTICS

Element Characteristics	Propulsion Element				
	FGB	SM	Prog-M	Prog-M2	ATF*
Propellant Capacity (kg)	6120	860	1740	3460	1760
Propellant Tank Residuals (kg)	360	60	60	80	80
Useable Propellant (kg)	5760	800	1680	3380	1680
Initial Propellant Load (kg)	5067	860	1740	3460	1760
Main Engine Thrust (kgf)	417	312	300	300	
Main Engine Isp (sec)	298	300	300	300	
ACS Thrust (kgf)	40	13	13	13	13
ACS Isp (sec)	252	250	250	283	290
Deploy Altitude (km)			220	220	
Orbit transfer Prop $\Delta P/\Delta H$ (kg/km)			1.4	2.6	
Rendezvous/Docking Prop (kg)			140	225	
Deorbit Target Altitude (km)			80	80	
Deorbit Transfer Prop $\Delta P/\Delta H$ (kg/km)			.6	.8	
Separation Prop (kg)			18	85	

NOTE: ATF = Autonomous Thruster Facility and is delivered on a dedicated Progress flight. However, the Progress flight does not deliver other logistics or additional propellant for the ISS to use (Puckett, 1995:8).

The propellant available from the Progress vehicle for the ISSA equals the useable propellant minus the orbit transfer, rendezvous, docking, departure, and de-orbit propellant. Puckett determines that the available propellant equates to about 1000 kg from a Progress M and about 2300 from a Progress M2. The available propellant left over from the reboost will be transferred from the Progress to the FGB, first, and then to the SM. Puckett also declares that the tanks on the SM will be re-filled from either the FGB or the Progress if a) there is not sufficient available propellant to complete the

maneuver, or b) the SM is below 40% of its maximum capacity. He also assumes that no more than six Progress vehicle launches are available.

Because STRAP uses a constant interval to schedule reboosts, Progress vehicle flights do not vary, except by slipped flights. The problem posed by this fact occurs when other vehicles and micro-gravity requirements factor into the schedule. Currently, the Traffic Model inputs the results from STRAP and determines if the schedule is feasible. If it is not, an iteration process between STRAP and the Traffic Model occurs until a feasible schedule is found.

2.6 The Traffic Model

A traffic model looks at the details necessary for Space Station Operations. It supports the development of robust long-term space station planning to maximize utilization of the station's assets within Program constraints (ISS Traffic Model: 1-6). The Traffic Model (TM2) calculates values to determine a feasible schedule that incorporates the details of successful operation. Some of these details include station reboosts, crew, crew logistics, maintenance supplies, international flights, and quiet periods of micro-gravity where no docking/undocking is allowed. Lemmons (1995) also lists many of the assumptions and requirements, or planning allocations, involved in the traffic model.

Not all of these assumptions or requirements are relevant to the present research, therefore Table 2 gives a brief summary of the requirements deemed pertinent to this study.

TABLE 2

APPLICABLE REQUIREMENTS FOR RESEARCH

A	RSA will provide all of the propellant via Progress vehicles.
B	No more than 6 total Progress M and M2 flights per year
C	The SM and FGB can supply propellant directly to the Progress.
D	Russia will provide supplies for a flight crew of three and the US will provide supplies for a crew of four.
E	The Space Shuttle launches all US on-orbit segment logistics, maintenance, and utilization requirements.
F	Russian vehicles launch Russian segment logistics, maintenance, and utilization requirements.
G	Space Shuttle cargo loading priority: 1. Utilization 2. Crew Supplies 3. Logistics and Maintenance
H	Russian vehicles cargo loading priority: 1. Crew Supplies 2. Water 3. Extra Vehicular Activity 4. Gas 5. Propellant 6. Logistics and Maintenance 7. Utilization
I	No Progress or Soyuz docking/undocking while the Space Shuttle is docked
J	Minimum of 2 days between Russian vehicle undock date and next Russian vehicle launch date.

(Lemmons, 1995)

2.7 Simulation Modeling

Modeling is a tool used to abstract, represent, and analyze systems, processes, and proposals. Simulation models can take the form of a physical model, mathematical model, and/or computer model. Law and Kelton (1991) define a system as a collection of items which act and interact to the accomplishment of some logical end (3). Models are

used for various reasons. Some people use models because they are more cost effective than running tests on the real system. In addition to expense, tests may also involve unacceptable risk or item destruction. Others use models because a situation cannot be tested in real life, for example, wargaming versus a real war. Simulations can be used to help analyze designs of systems that do not yet exist. Simulations can also be used to isolate conditions to evaluate outputs or even to study systems over period of time.

Pritsker (1986) specifies that simulation modeling may be used at four different levels (Pritsker: 1). The first level is as a device to explain or define a problem. Analysis is another use and can help identify important problems, issues, situations, and critical factors. Simulations also may be used in design to help assess different options. Finally, Pritsker mentions that simulations can help decision makers look at various predictions for future plans.

The use of simulation in this research has been a combination of all four areas suggested by Pritsker. We first used simulation to define the various elements of the problem. Once the problem was defined on a basic level, we identified issues that may be of interest to NASA and show how these parameters may be isolated for close analysis. Different altitude strategies will be compared and will visually depict how these affect the decisions to be made. While our simulation is not a basis for future plans, we use it to describe how different approaches can aid in planning.

Many different types of simulation models exist. Some common simulation models include *continuous, discrete, static, dynamic, deterministic,* and *stochastic.* These

models use variables which characterize the state of a system. Law and Kelton define two types of systems, discrete and continuous (6-7). *Discrete* models have systems where the state variables change instantly when certain events occur. *Continuous* models change the state variables continuously with respect to time. Models which represent a system at a certain time, or in which time in not considered, are known as *static* models. Law and Kelton contrast static models with *dynamic* models, which portray systems as they evolve over time. The inclusion of random components differentiate between the last two model types. *Deterministic* models do not have any randomness associate with them while *probabilistic* models do contain probabilistic components.

Hartman (1985) confines the definition of a model by stating that models imitate the important characteristics of a real system in order to describe or predict the behavior or outcome. Since no model can depict everything about a real system, Hartman uses the validity of a model as an evaluation method. He states that the validity of a model depends on the structure of the model and the intended use. According to Hartman, a simulation model acts out the interactions of a system and are useful for models with procedures instead of (or in addition to) formulas. He also declares that simulation solution methods work well with dynamic models. Sensitivity analysis is accomplished through repetitive runs with changed inputs. Sometimes the only change in inputs is by using different random numbers.

NASA often works with different types of simulation models. These models can be broken into at least two different categories. The first category consists of physical

models of systems. One example of this type of model is NASA's KC-135 airplane, affectionately nicknamed the Vomit Comet, that allows for twenty seconds of free-fall to simulate a weightless environment (*Popular Mechanics*; 16). The free-fall environment results from the KC-135 flying in parabolic paths.

The second category consists of computer-based simulations. Most of these computer simulations use engineering equations and fixed parameters to model different systems. A computer simulation that simulates the environment of a inflatable structure on the moon is one example of a discrete model simulation. This system is used to study the structure under a variety of loads for a fixed set of time independent, non-random conditions. NASA's STRAP and Traffic Model (see section 2.5.1 and 2.6, respectively) are also good examples of computer-based simulation models with engineering equations and fixed parameters.

A third type of category is a simulation model which incorporates randomness. Usually randomness is achieved through the use of random number generators to determine various input parameters. Models which utilize random numbers are commonly known as *Monte Carlo Simulations*. Some authors, like Law and Kelton, restrict this type of simulation to be a scheme which employs random numbers and is used for solving problems where time plays no substantive role (Law: 113). For example, one might wish to characterize composite probability distributions by repetitions combining the underlying distributions in order to create histogram shapes and calculate other distributions parameters. We however, use this term to indicate a

simulation model which uses randomness in either a time-based or non-time-based context.

When using a simulation, one must take care to only use the simulation for the purpose it was created. There exists a natural tendency to extrapolate information and project it for uses beyond the intended design. This practice is dangerous. Simulation models are abstractions of a system, many simplifications and assumptions used to construct a particular abstraction for a particular purpose do not often apply beyond that designed purpose.

3. *General Methodology*

3.1 Chapter Overview

In this chapter we describe the problem and introduce the Iterative Approach, which is the logic we followed for the development and the execution of this thesis. We provide various definitions, assumptions, and examples that provide the basis for the details of Chapter 4.

3.2 Statement of the Problem

NASA/OC is responsible for the creation of operational plans to ensure the mission of the space station is accomplished over its intended lifetime. These operational plans include monitoring solar activity, choosing altitude strategies, generating altitude profiles (including reboost strategies and propellant requirements), and creating traffic model schedules. Currently the method is a complex iterative process, using various computer models, file transfers, and manual updates until a fixed schedule is converged upon. The situation is looked at from a long term engineering design perspective.

The purpose of this research was to provide NASA/OC with a fresh approach of looking at situations relating to the ISS and to demonstrate how analytical tools used in Operations Research may be implemented to aid in shifting the fixed parameter, dynamic operational design approach to a probabilistic modeling approach. By altering the view

of the situation, NASA will gain a different perspective that could improve the quality, ease and speed of their space station operational planning processes.

3.3 Rational

NASA's Operational Planning Team creates actual operational plans from methods developed by the Design Analysis Team. The Design Analysis Team instigated a series of computer programs to determine the feasibility of proposed decisions and allow the results of decisions to be quickly observed. Our first step created prototype models representing the iterative flow of data through NASA's series of programs. Figure 3.1 illustrates our view of the flow of data through NASA's programs while Figure 3.2 presents our prototype models.

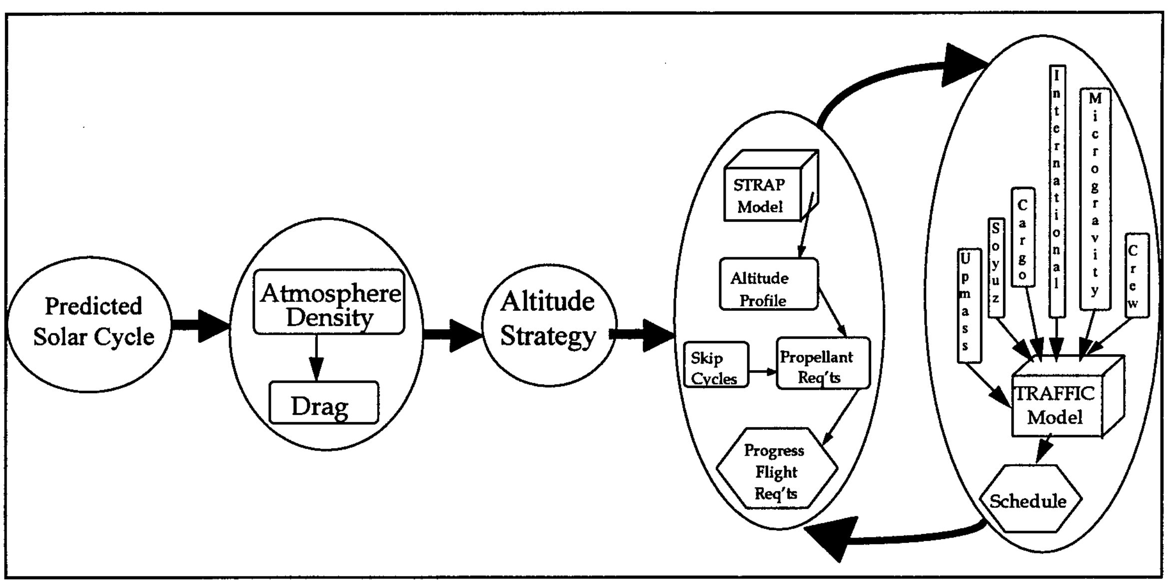


FIGURE 3.1 Detailed Flow Chart of Data in NASA

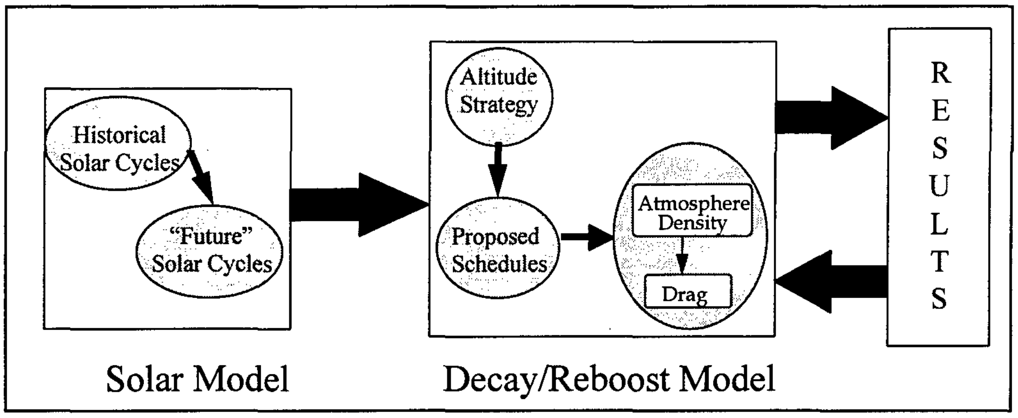


FIGURE 3.2 Flow Chart of Data in Prototype Models

Our models of this process were not intended to replicate or replace the actual flow, but to demonstrate related concepts on simplified, aggregate examples. We wanted to visually identify the status of several key issues over a spectrum of the possible outcomes resulting from the randomness inherent in the system. Key issues include aspects of the scenario that could be of interest to decision makers. Suggested parameters of interest may include:

Given a large number of runs of a particular altitude strategy (each run corresponding to different random outcomes):

1. What percentage runs or years violate the required 180 days of microgravity per year?

2. How many reboosts were required to keep the ISS at a functioning orbit?

3. Calculated at regular time intervals, if reboost support from new launches was indefinitely postponed . . .

    a. How many days will the space station remain in orbit?*

    b. Over all these interval calculations, what is the resulting minimum orbital lifetime before reaching 278 Km under nominal conditions?

*NOTE: NASA requires the on-orbit Space Station shall have sufficient skip-cycle reserve propellant for a reboost to an altitude that results in at least 360 days of orbital decay to 278 Km under nominal operations (Puckett, December 1994: 9)

4. In a fixed schedule, which launch windows are the most sensitive (or least flexible)?

5. What is the average fuel on board?

Given a chosen altitude strategy, outcomes can vary due to the different random components inherent in the system. For example, levels of solar activity, launch slips and mission cancellations cannot be predicted, except in long term averages or distributions, because any one or more of the infinite combinations of these states could occur. We decided to model this random behavior of the system by running a simulation that draws random levels of solar activity, slips, and cancellations from set distributions. Over many runs, the outcomes of the simulation formed a history of the various parameters of interest and, using pictorial graphs such as histograms, results could be easily viewed. While we created prototype representations of all models required to complete the loop of data flow represented by Figure 3.1, the area of our concentration was on altitude profile strategies which were robust enough to encompass the random nature of solar activity. The following section describes the basic logic of the procedure and defines the terms we used.

3.4 DEFINITION OF TERMS

This research consisted of a **Phase Approach**. The Phase Approach endeavored to start the modeling of a process with a general idea and to present a step by step methodology to improving the model, similar to the first steps in a NASA Program/Project Life Cycle Process Flow on a basic level (see Figure 3.3) (Shishko: 23).

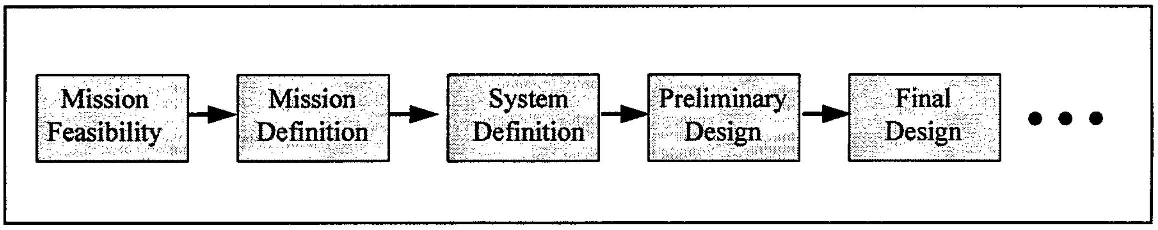


FIGURE 3.3 Basic NASA Program/Project Life Cycle Process Flow (Shishko: 23)

We began by defining thesis (mission) objectives, performance requirements, and customer needs. Then, we set up requirements to meet our objectives and established general measures of effectiveness, which were presented in a Thesis Proposal. The preliminary design consisted of a very simple model designed to close the loop of data in the operational planning process. From this initial design, recommendations for improvement emerged and were incorporated in the next phase. This methodology allows the important aspects of a process to be identified and included, albeit on a basic level, to receive output. The output then points to aspects which need to be improved in detail, often in a hierarchical manner. As the methodology is followed, the procedure may be stopped at the end of each of the phases and general analyses of the output can be made. This approach continues until either no more improvements can be made or the decision

maker decides to stop. Figure 3.4 gives a visual representation of this idea: Hopefully, each phase moves the process further down the funnel to the desired final design.

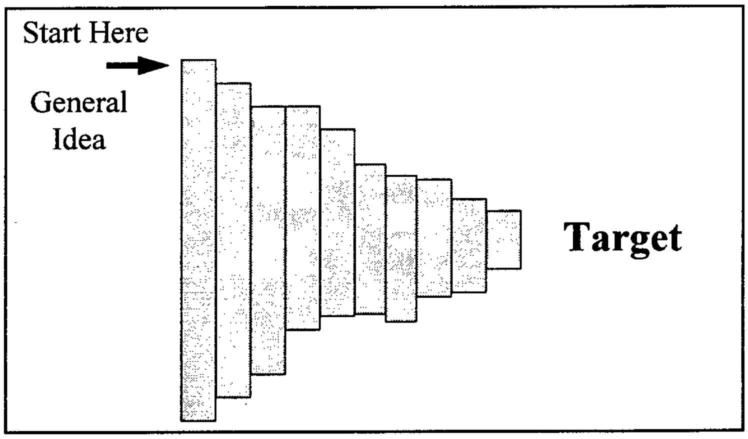


FIGURE 3.4 Funnel Example of a Phase Approach

In our research, Phase I made the loop of data transfer for the operation plan as simple as possible. In this phase, many assumptions were be made and most variables, those values yet to be determined, were be fixed. Subsequent phases removed various restrictive assumptions and allowed more variables to alternate.

Within each phase, **experiments** were made. One experiment consists of a statistically significant number of computer simulation **runs** for a chosen altitude strategy. Each run produced various output data for a candidate future of the fifteen-year station life. These data included predicted altitude profile, ISS onboard fuel levels, microgravity blocks, reboosts, and a feasible schedule. By feasible we mean a deconflicted schedule of docking, undocking, and reboosts. In the early phases, a feasible schedule did not imply that microgravity requirements were met. In later phases, feasible also meant launch slips and cancellations. Future phases, past this research, could directly schedule microgravity as well. The schedule varies among the runs because of

the random draws from the distributions for solar activity, and between runs, depending on which variables are allowed to change.

3.5 General Assumptions

While each phase had its own set of assumptions, general assumptions exist which applied to all phases. We first assumed the time frame we are concerned with is after the station has been constructed. Assuming space station assembly launches will begin in 1997 and end in 2001, our study starts at the beginning of 2005. We used a zero-order approach to the prediction of the solar cycle. This approach assumes that the next solar cycle is independent of the last solar cycle. In addition, we assumed that the initial conditions of a run included: only a Progress M2 attached, fuel storage tanks are nominally half full, and the station was at a lifetime altitude of 360 days to 278 km. This lifetime is a NASA requirement states that the Space Station needs to maintain sufficient skip-cycle reserve propellant for a reboost to an altitude that results in at least 360 days of orbital decay to 278 km under nominal operations (Puckett, 1994:9). We refer to this lifetime altitude as **T**hree-sixty **T**o **T**wo-seventy-eight, or T^3. The last general assumption, found in the following paragraph, came from Cargo prioritization.

Cargo prioritization is listed in the TM Version 2.0 Requirements (TM2). Here propellant is listed fifth in the order of priority (Lemmons, 1995: 1-8). Therefore, we define the first four elements, crew supplies, water, EVA (Extra Vehicular Activity), and gas, as **critical supplies**. **Nominal supplies** consist of maintenance and logistics, and

utilization. Propellant needed for Progress vehicle launch, docking, undocking and rendezvous is **critical propellant**. The **nominal propellant** is fuel allowed first for the reboost, and second, left over that can go into the ISS storage tanks. We have assumed that critical propellant can replace nominal supplies if the capacity is needed. Nominal propellant fell last on the list of priorities. If extra capacity exists after critical supplies, critical propellant, and nominal logistics are accounted for, then nominal storage propellant will be launched as storage tank capacity permits (See Figure 3.5).

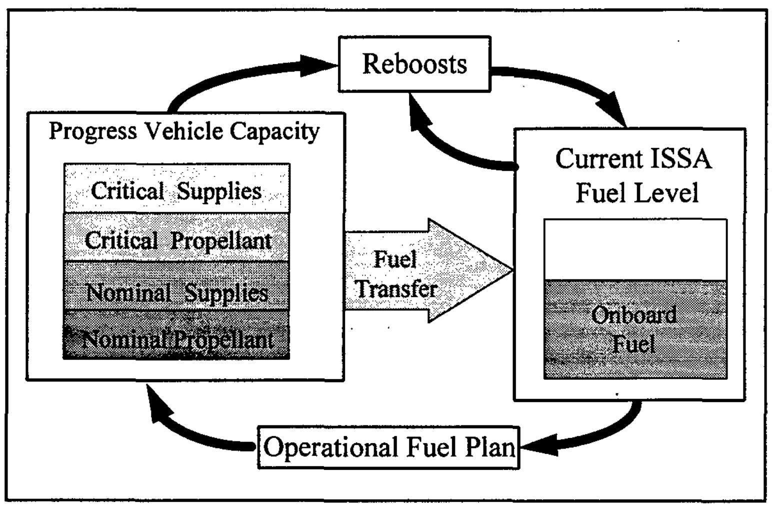


FIGURE 3.5 Propellant Requirements

We also assumed that fuel from storage can be transferred to a Progress vehicle docked to the ISS for a station reboost. Other assumptions were added as the research continued.

3.6 PHASE I OF METHODOLOGY

Phase I modeled the process in the lowest form of detail to complete one experiment. In this phase we assumed that all flights other than Progress vehicles were fixed in the schedule. Maintenance blocks were not planned in the schedule and reboosts occurred in one day. In addition, we assumed that no slips or cancellations of the Progress were allowed. The only random variables were the values of solar activity and the proportion of capacity allotted to nominal propellant. Additionally, microgravity time block requirements will be reported, but will *not* be scheduled, as this was one of the variables which we demonstrate changes as uncertainty increases. Step 1 of all phases randomly created a future solar cycle consisting of two historical solar cycle shapes with randomly chosen maximum height, length, and daily deviations. This process is described in detail in Chapter 4 (See Figure 3.6).

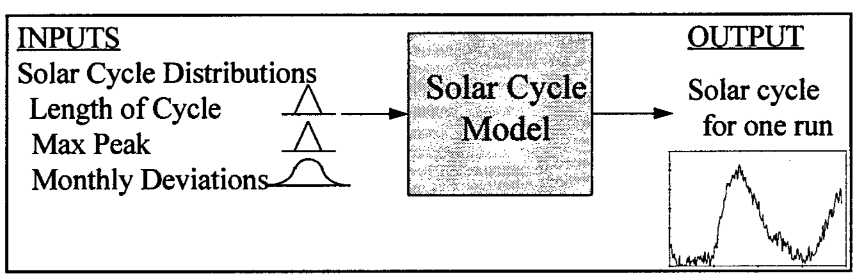


FIGURE 3.6 STEP 1 Solar Cycle Model

One way to think about the computer simulation is to divide it into two separate sides. The first side consists of the *true* future values for the solar cycle (sometimes called *God's-Eye View*). The true values are unknown, and will remain unknown until the actual time of occurrence. Although these values are not the real values which will be

observed in the future, this side models the actual data that would be observed at the time of occurrence. The simulation drew these values for the *real* cycle at random from the various distributions (i.e. one run), used them for daily drag computations, and hence, determined the *actual* altitude profile over the life of the station for each run. This *true* side creates a reality that the other side of the simulation actually performs against. The other side of the simulation did not know these values until the day they occurred, but was forced to predict them in order to conduct operational planning. This side models the reality of not knowing the value of solar activity, created by the *true* side of the simulation, until the day it is observed. During Phase I, this side of the model had perfect knowledge of the upcoming solar activity to generate a single schedule for the life of the station.

In Step 2 of Phase I we chose an altitude strategy and used this strategy for all runs. We began with a basic strategy that would reboost the station when it reached a designated altitude. Only one strategy was chosen for Phase I (hence, only one experiment in Phase I). See Figure 3.7 below.

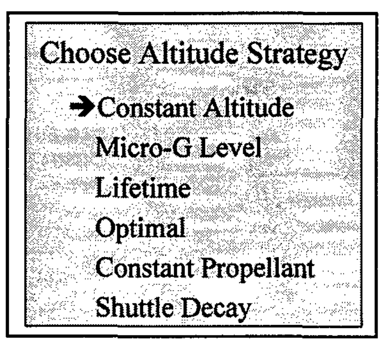


FIGURE 3.7 STEP 2 Altitude Strategy Model

Step 3 of the first phase took the given altitude strategy from Step 2, the value of solar activity received from Step 1, the initial altitude, and scheduled flights arbitrarily and input them into the Altitude Profile Model (see Figure 3.8). Based on nominal decay predictions, an ideal altitude profile was generated to include projected reboost dates.

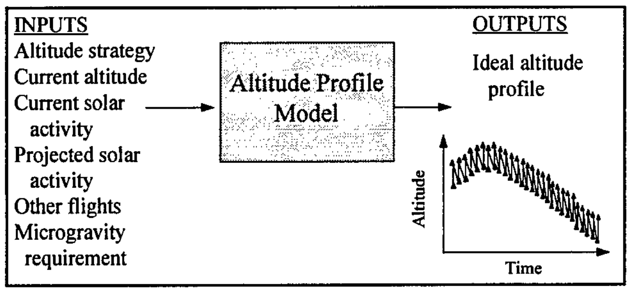


FIGURE 3.8 STEP 3 Altitude Profile Model

In Step 4 of Phase I we compared the desired reboost date with the other vehicles in the fixed schedule in an attempt to meet the ideal altitude profile. If there was a docking conflict, the program automatically moved the reboost later in the schedule, to the first free day. If not, the reboost was scheduled (see Figure 3.9).

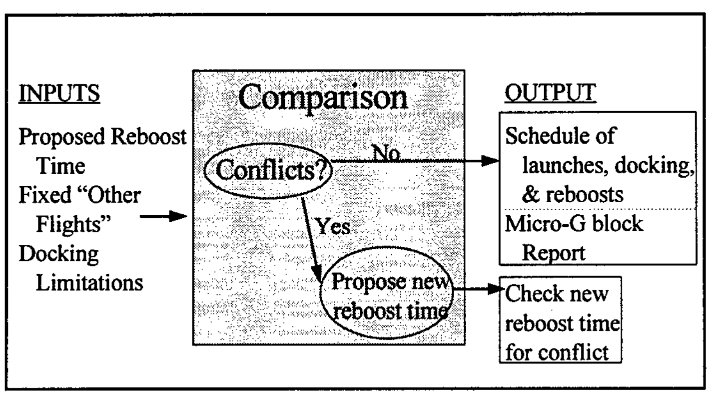


FIGURE 3.9 STEP 4 Decision Tree

Important parameters such as days of microgravity per year and number of reboosts per year, from each run were plotted on histograms. These histograms allowed

us to view the resulting distributions of different issues of interest based on the random outcomes achieved by the simulation model. For example, one histogram contains the total lengths of microgravity blocks in each run. Once this phase is completed, we will move onto the next phase.

3.7 FUTURE PHASES

Each phase attempts to move down the model and improve the models and the strategy used in operations. Limiting assumptions were removed and the models improved in realism. Chapter 4 describes the details of these phases and Chapter 5 includes potential phases for future research and various research tools which may be implemented to improve operations.

3.8 CONCLUSION

While this research effort endeavored to enlighten NASA on various approaches to looking at problems relating to the ISS, it does not provide NASA with a handbook of tools or step-by-step instructions to use. We chose to limit our research to simple representations of different phases of the NASA/OC effort and illustrating some new approaches to different areas. After replacing their large, detailed models with aggregate, simple models, the next sections show how selected Operations Research tools can help improve the decisions made through new approaches and ways of looking at the problems. Although we present NASA will different possible tools, such as scheduling,

assignment, and transportation, that could eventually be implemented, Simulation Modeling was our primary tool in this research. We provided NASA with a detailed summary of all models used including any assumptions made, model limitations, how the model transfers data, interaction diagrams, and computer language/platform/runtimes, output histograms (see Figure 3.10).

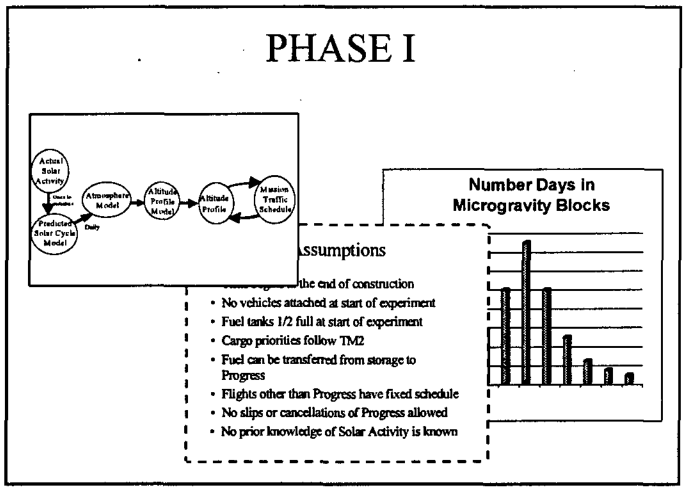


FIGURE 3.10 Example of Phase I Experiment

While this simulation does not directly transfer to the large detailed models, it hopefully gives NASA an insight into available Operations Research methodologies and give them a starting point for future research into modeling approaches that can help NASA improve the speed, difficulty, and quality of their job. Unfortunately, because their models are so large and complicated, it will not be likely for NASA to find a specific *off the shelf* tool that will completely automate this decision making process. The end result is to present NASA with a different perspective that can help them evaluate their operational traffic planning by describing OR methodologies that can explicitly encompass randomness in the system.

The next chapter describes the details of this general methodology and presents results from the many runs of the different strategies.

4. *Detailed Methodology and Results*

4.1 Chapter Overview

Chapter 4 develops the methodology presented in Chapter 3 to a greater detail. That previous chapter introduced the Phase Approach and described how we wanted to create simple, yet representative, models of the flow of data found in NASA's Design Analysis Group. Our first goal was to complete the flow of data as quickly as possible, at the most basic level. The phases all needed to contain a model that portrayed realistic values of solar activity in a manner typical of the sun. All phases also had a set of rules to follow to determine what to do, when, and how much. This set of rules is known as an altitude strategy and determines if it was time for vehicles to dock, undock, or reboost. If a reboost, the altitude strategy helps determine how high to reboost and how much propellant to use. Acquiring statistics on our parameters of interest required collecting daily, monthly, and sometimes yearly values. These statistics gave us the information we desired to compare various altitude strategies and other important decisions.

4.2 Solar Model

We began the flow of data by creating Solar Model. The Solar Model was an attempt to imitate, in enough detail to be useful, the levels of activity from the Sun. The Zurich index is one of the most common indicators of solar activity. This index measures the number of sunspots and sunspot groups observed on the sun and takes into account a correction factor for observation error. In 1852 Heinrich Schwabe discovered that, on the

long term yearly average, the sunspot number displayed a cyclical nature with a period of seven to thirteen years.

Our initial attempt at this model was to develop an equation to predict these sunspot numbers using curve fitting and splines. This method proved to be extremely complicated and very inaccurate. While some people may be able to use these standard statistical techniques, we abandoned this approach.

Next, we looked at the past values and plotted these historical values to get an idea of the *shape* of the cycles. Figure 4.1 contains a graph of these historical shapes:

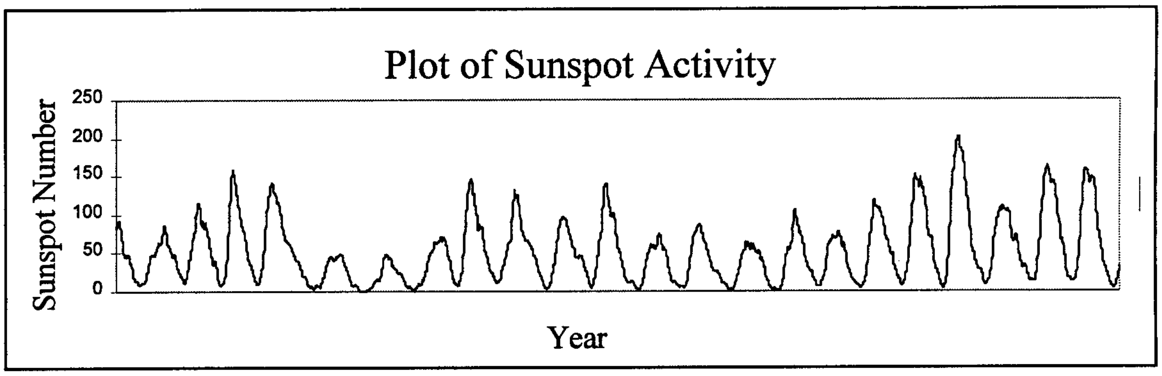


Figure 4.1 Historical Shapes of the Solar Cycle (Thompson, 1995)

The numbers used for this plot came from Richard Thompson's smoothed sunspot values (see Chapter 2.1). Our desire was to create a solar cycle which had similar characteristics as the historical cycles. One way to do this would be to randomly pick one of the historical cycles and assume a future cycle will follow this basic shape but possibly differ in length and/or height. However, using the historical cycles directly limited us to twenty-two historical cycles and we wanted to simulate hundreds of cycles. To directly compensate for this prior knowledge, we decided to combine *two* historical cycles and

assign a random weight to each of the cycles. Although none of these cycles will be the actual future cycle, for illustrative purposes, we will see behaviors which closely depict what future cycles may look like over enough runs. In order to prevent combining two low cycles or two high cycles, we standardized all the shapes to equal lengths and heights before combination and then randomized the height and length of the new cycle.

Starting with Thompson's smoothed sunspot values, we broke the data up into sunspot cycles instead of years. Since a solar cycle is defined from minimum solar activity to minimum solar activity, we assumed a cycle ended when the values stopped decreasing and began increasing. If there was more than one low value, we split the cycle between the values. If the number of low values was an odd number, we gave the remaining one to the previous solar cycle.

After breaking the data up into cycles, we standardized the data. Standardization allowed the historical shapes to compete on an equal basis for combination. We wrote a FORTRAN program to transform the cycles into equal lengths of 132 months (11 years), which is a rounded average cycle length (see Appendix A). We then inserted new lengths into Excel and standardized to peak heights of 114, which is the rounded average of the peak heights of all the cycles (see Appendix B). The data, now standardized to common lengths and heights, were then stored in a file for use in the simulation of the solar model. The following figure presents two of the twenty-two standardized historical shapes.

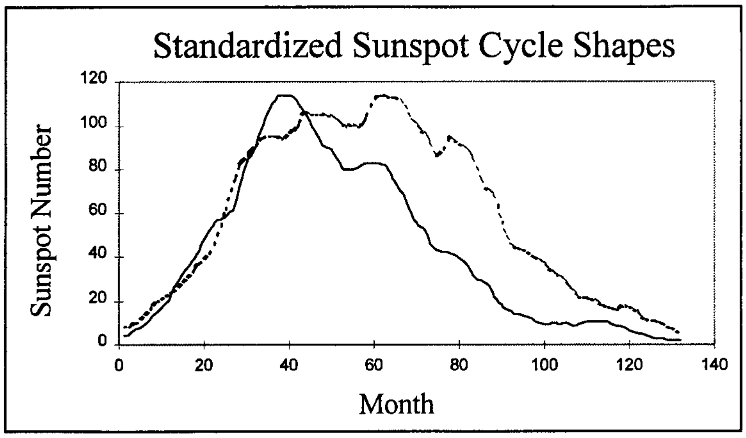


FIGURE 4.2 Graph of Standardized Historical Solar Shapes

The Solar Model read in the standardized solar cycle data and randomly chose two streams. We then combined the two streams in a near convex combination of length and height (see section 4.1.1 and 4.1.2). Figure 4.3 shows the two streams from Figure 4.2 combined. Note that, because the peaks occurred at different times, the maximum height of the new shape is less than the standardized shape of 114.

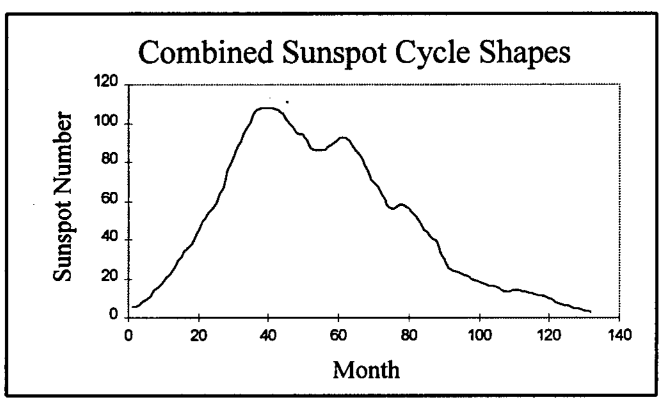


FIGURE 4.3 Graph of Combines Streams

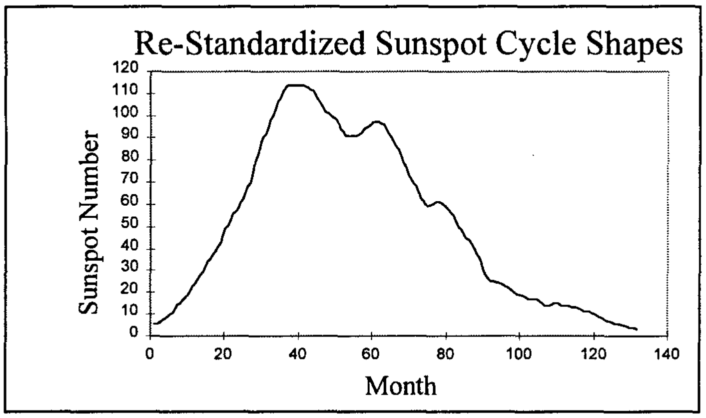


FIGURE 4.4 Graph of Re-Standardized Combined Streams

After combining the two cycles, we re-standardized the height of the cycle to the value of 114 (see Figure 4.4). This combined, standardized cycle was now ready to undergo variation in height, length, and daily deviations. From the random number generator and the standard height and length, the program draws varied the height and the length of the stream (see sections 4.2.3 and 4.2.4). Monthly deviations from the given value produce the fluctuations commonly seen with solar activity (see section 4.2.5). The final stream is one that has a shape of two historical streams, but which varies in height, length, and monthly deviations:

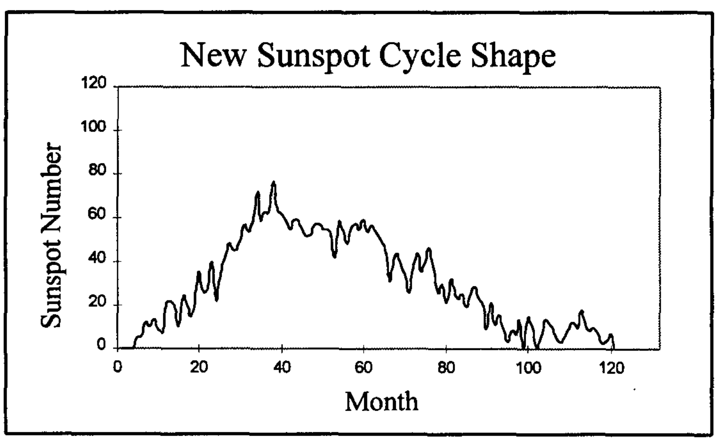


FIGURE 4.5 Plot of Final Stream

We then augmented three of these streams and took the tail of one, a full cycle, and the beginning of the last. The extended solar cycle consists of one full cycle, the tail of the previous cycle, and the beginning of the next cycle (see section 4.2.6):

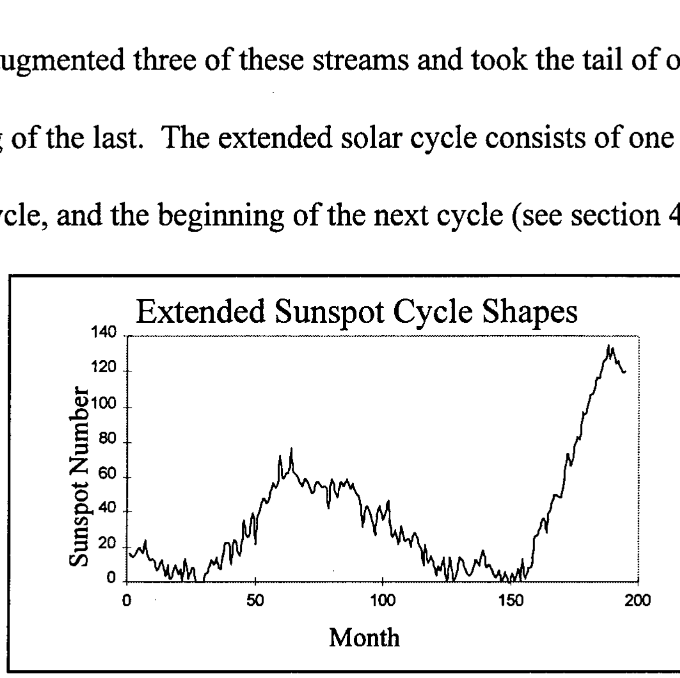


FIGURE 4.6 Plot of Extended Solar Cycle

To show some possible different behaviors, we created many solar cycles using new sets of random numbers for each extended cycle. This data was placed in arrays and then printed to files. One file was for input into Excel and another file was for input into the Decay/Reboost Model.

4.2.1 Convex Combinations

Convex combinations assure that all desirable representative values lie strictly between the two given values. The following is the standard equation for a convex combination:

$$x_{cc} = \alpha \cdot x_1 + (1-\alpha)x_2 \qquad \textbf{(4.1)}$$

where $0 \le \alpha \le 1$

For example: when $\alpha = 0$ then $x_{cc} = x_2$

$\alpha = 1$ then $x_{cc} = x_1$

$\alpha = .3$ then $x_{cc} = 0.3x_1 + 0.7x_2$

This standard equation can be expanded to include more than two values. In this case, the rules governing a convex combination requires all the weights to be between zero and one and the sum of all weights must equal one:

$$x_{cc} = \alpha_1 \cdot x_1 + \alpha_2 \cdot x_2 + \ldots + \alpha_n \cdot x_n \tag{4.2}$$

where

$$0 \le \alpha_i \le 1 \qquad i = 1,\ldots,n$$
$$\sum \alpha_i = 1$$

When we took a convex combination of two shapes, we expanded on the idea of the standard convex combination and combined two shapes to form a new shape. Each of these shapes had a randomly generated weight attached. While the new stream is a combination of the old streams, a convex combination bounds the feasible region of the new stream by the values of the old stream. We cannot make this assumption since our historical data does not necessarily represent the extreme boundaries of possible solar cycle shapes. Rather than bounding the feasible region of our solar cycle by the region set by the old cycles, we increased the allowable region by 4.7 percent (see section 4.2.3). For this reason we call the combination a *near convex combination* of shapes.

4.2.2 Expanding the Feasible Region

Using order statistics in the methodology outlined below, we expanded the solar cycle shape region by 4.7 percent. To obtain this value we first took the twenty-two standardized historical solar cycle shapes and overlaid each of the shapes to contrast the monthly values for each 132 months. At each month we ordered the values from the

smallest value (Y_0) to the largest value (Y_n) of the twenty-two cycles. To allow for the potential increase beyond the boundary imposed by Y_n and Y_0 we estimated the next value in the sequence, Y_{n+1}, using the following equation:

$$Y_{n+1} = \frac{(n+1)\cdot Y_1 - (n^2+n)\cdot Y_n}{1-n^2} \tag{4.3}$$

where

Y_{n+1} = estimated value of the next number in the sequence

n = number of values in the sequence

Y_1 = smallest value in the sequence

Y_n = largest value in the sequence

To use this equation we needed to assume that all of the twenty-two data points in each month were uniformly distributed. After estimating the next value, we found the ratio Y_{n+1}/Y_n for each month. We took the maximum ratio of all 132 months and obtained the percent increase using the following equation:

$$\text{percent increase} = 100\cdot\left(\frac{Y_{n+1}}{Y_n}\right) - 100 \tag{4.4}$$

The percent increase allowed the feasible region of our solar cycle shapes to permeate the original boundaries and was used in conjunction with the random number generator to combine historical cycle shapes.

From the historical cycle shapes, we randomly chose two for combination. Using the random number generator, the we drew a uniform(0,1) random number for the weight

of the first cycle shape. Next, we multiplied the weight by the value 1.047 (a 4.7 percent increase). This multiplication gave the first cycle shape a weight, theoretically, between zero and 1.047. The weight of the second cycle equals one minus the weight of the first cycle. Since all of the solar cycle shapes have an equal probability of being chosen first and second, no bias is induced by always giving the first cycle the larger weight. This combination differs from a convex combination in that the weights are allowed to be greater than one and less than zero. The sum of the weights, however, still equal one. The new solar cycle may now permeate the boundaries of the feasible region.

4.2.3 Random Variation of the Height

To vary the height of each cycle, we first determined the distribution of the peak heights of the twenty-two historical cycles. After some preliminary analysis, we decided to test the Triangular distribution, as we only have twenty-two cycles of data. This distribution provides a good rough model in the absence of data (Law: 341). To test this distribution, we used a common statistical test, the Chi-Square Test. The Chi-Square Test allows us to test how well the data fits a hypothesized distribution. We began by hypothesizing that the data followed a Triangular distribution with a low value of 40, median value of 77 and high value of 205. After creating and running Chi-Square Tests, we determined that the Triangular distribution best represents the variations in the peak height of the solar cycles.

After deciding to use the Triangular distribution with parameters 40,77, and 205, we drew from this distribution to determine the peak height of each cycle. To correspond to the peak cycle height, we transformed each of the data points in the cycle.

4.2.4 Random Variation of the Length

Using the same techniques as for the height, we determined the distribution for the length. For the length, we hypothesized the data followed a Triangular distribution with parameters 105,119, and 172. After running the Chi-Square test, we concluded that the data fit a Triangular (105,119,172) distribution. From this distribution we drew a random length for the cycle. Once the new length of the cycle was determined, the 132 points in the standard cycle were transformed to fit into the new cycle length.

4.2.5 Random Monthly Deviations

Once we have the new cycle height and length, we gave the cycle random monthly deviations to obtain realism. We obtained the daily deviation by drawing from a clipped Normal distribution with a mean of the given value and a standard deviation of five. We chose the Normal under the assumption that most of the time the actual value will cluster about the mean but could deviate either way with equal probability. If the draw resulted in a number below zero, the sunspot value was given as zero. The new cycle was now transformed to a new height, length, and given monthly fluctuations. The next step was to create an extended solar cycle to allow a longer simulation.

4.2.6 Creation of an Extended Solar Cycle

We repeated the process of creating a solar cycle three times. A uniform random variate of one to three years (twelve to thirty-six months) decided the length of the *tail* of the first cycle. The tail includes 12 to 48 months of data from the previous cycle. Only the information concerning the tail was kept. The second cycle was used in its entirety. As we need over 16 years of simulated solar activity (195 months), the beginning length of the third cycle was:

$$195 \text{ months} - (\text{length of first cycle's tail} + \text{length of second cycle}) \qquad (4.5)$$

We kept only the information of the beginning of the third cycle and discarded the rest. The tail of the first cycle, the second cycle, and the beginning of the third cycle were augmented to form an extended solar cycle (or solar cycle).

We created many different extended solar cycles to capture the possible range of behaviors for the next solar cycle.

4.3 Decay/Reboost Model

The Decay/Reboost Model was the second model in the loop. This model reads in the different three-cycle worlds from the file created by the Solar Model and stores the monthly solar data in an array. The data is read in as the 12-month smoothed Zurich sunspot number and then converted to the solar flux value using NASA's equation (TM-82478). A loop allows running many extended solar cycles. The number of reboosts completed in a world, or 15 years, begins at zero, as does the "waste" propellant. Waste propellant is propellant from the Progress vehicle that exceeds the capacity of the space

station. This propellant is thrown overboard. Since this will never happen in reality, this strategy can be used as a measure of how infeasible a strategy is.

4.3.1 Phase I

Phase I consists of our most basic altitude strategy. The initial altitude begins at a uniformly random number between 300-370 km. The density at the altitude is calculated daily and the decay lifetime every 5 days. The altitude strategy we are using in this model is based on the lifetime altitude of at least 360 days of orbital decay to 278 km under nominal operations [13:9]. We refer to this lifetime altitude as Three-sixty To Two-seventy-eight, or T^3. If the decay rate is less than or equal to 278 and the twenty day between boost constraint is not violated, the station will be reboosted. A Progress M2 will be used for the reboost. This model assumes that a Progress will instantly appear, dock, and reboost the day it is needed. We specified twenty days as the minimum number of days between reboosts . At this level of detail, it is not yet necessary to model the launch and time to station of the Progress. The reboost will be accomplished in an attempt to use all available propellant. We created an iterative scheme to determine the amount of propellant to use, with a given tolerance between the desired and the actual. If the reboost reached 460 km, the maximum allowable altitude based on Russian hardware design constraints, the reboost terminated and any remaining propellant (after filling onboard tanks to capacity) was cast overboard. Waste propellant was calculated through the lifetime. The daily altitude, waste propellant, and number of boosts were kept as statistics.

4.3.1.1 Density Calculations

While density is the key input to orbital decay, calculating it is extremely difficult because of the many parameters affecting it. Solar activity, altitude, local time, and latitude are just some of the contributions that can cause the density to vary. Because of the uncertainty associated with calculating density, all attempts to predict the density of the atmosphere for an object in orbit are only rough estimates. Altitude and solar activity are the most commonly used parameters. Usually models divide solar activity into two or three categories ranging from low to high levels of solar activity. Altitude often steps in blocks of fifty or one hundred. Some models include nighttime minimums and daytime maximums.

For our research we compared two different models. The first model was a graph relating atmospheric density to altitude. NASA plotted four different curves on this graph: daytime maximum at high solar activity, nighttime minimum at high solar activity, daytime maximum at low solar activity, and nighttime minimum at low solar activity (TM-82478, 2-10). The second model is a chart that splits solar activity into mean and maximum levels and then divides altitude into blocks of fifty. Interposed on a graph, the two models appeared to compare fairly well, however, the calculations are sensitive to small variations in the density and large variations may later result.

Our model for density mainly incorporated the first model and used the second model as a visual check. Although the altitude in this model varies from 100 to 1000 km, we only used the curve between 150 and 500 km, typical space station ranges. We tied

down the values of the curve at the endpoints and used a logarithmic equation to estimate the curve between these points.

4.3.1.2 Altitude Calculations

To calculate the daily altitude of the orbiting object, we used Wiesel's height equation [Wiesel: 85]. This equation begins with the previous height and subtracts a value that includes parameters such as the gravitational constant μ, the radius of the earth, the previous height, the ballistic coefficient, the atmospheric density and the time span.

4.3.1.3 Reboost Calculations

In the calculation of a reboost we desired, in this phase, to use as much available propellant to reboost the station as high as possible. The complexity of solving the rocket equation in terms of the new height instead of the needed propellant for a given height forced us to develop an iterative scheme to obtain a tolerance between the propellant we desired to use and the propellant needed for a given height. We initialized the new height to the initial height and increased the height by 0.1 km per iteration. After the reboost, any remaining propellant went into the storage tanks on the space station. If the tanks were full, excess propellant went overboard.

4.3.1.4 Lifetime T^3 Calculation

The T^3 calculation was basically the altitude calculation computed for 360 days. At the end of 360 days, the resulting height was compared against the requirement of 278. If the station would fall below 278 during any time prior to the 360 days, a reboost was

executed (provided more than twenty days passed since the last reboost). T^3 was calculated every five days due to computation time demands of daily calculations.

4.3.1.5 Vehicle Schedule

For this phase we scheduled the Space Shuttle and the Soyuz directly into the code. Five shuttles arrived every year, starting on day seventy and spaced seventy-two days apart. The shuttle remained connected to the space station for seven days and, during this time, no other vehicles could dock, undock, or reboost. Dockings and undockings interrupted microgravity blocks. For microgravity to be counted, it must occur in blocks of thirty days or greater. Two Soyuz vehicles arrived each year. The first docked on day thirty and remained connected for 180 days. The second rendezvoused seven days before the first undocked and remained for 180 days.

4.3.1.6 Statistics Gathered

Graphs prove to be a useful tool for observing, validating, and comparing data. This phase collected statistics on a number of parameters. The height of the ISS plotted against time, in days, often pointed out coding errors or important criteria we forgot to include. A quick glance at this graph provides instant information and insights to the behavior of the station over this simulated run.

Other useful statistics included the maximum number of reboosts per year for each run and the smallest number of days of microgravity per year per run. These statistics were good indicators for how well the altitude strategy was working. For example, if one run had a maximum number of reboosts per year of seventeen, and the

maximum number allowed is six, then this strategy is not a good one to use. The number of days of microgravity per year is highly correlated to the number of reboost per year.

4.3.2 Phase II

From the results of phase one, we observed that the strategy did not work well in high solar activity. Some years there were zero days of microgravity blocks due to the extremely large number of required reboosts. For this next phase, we increased the number of days between reboosts to a minimum of sixty days. Also, phase one traded most of the onboard propellant for high altitudes and did not allow the buildup of the tanks. This high altitude does not optimize shuttle upmass. Our new altitude strategy reboosted the station to an altitude resulting in 360 days to 278 km without any onboard propellant. We also included two additional parameters of interest to compare strategies.

The first parameter was a penalty given for upmass lost on the space shuttle due to higher a higher rendezvous altitude. A rule of thumb given by McDonald and Teplitz is an additional 100 pound-mass per nautical mile lower rendezvous altitude (or, about 54 pound-mass per kilometer) (McDonald: 4). We noticed on example spreadsheets that the shuttle did not dock below 358 km. Therefore, we observed the altitude where the shuttle was docking during our current strategy. If the shuttle docked higher than 358 km it received a penalty. For instance, if the shuttle docked at 360 then we lost about 108 lbm of cargo. On the other hand, if the shuttle docked lower, then we gained more cargo. To prevent the constant use of all the onboard propellant, we limited the reboost to just above the T^3 calculation (see section 4.3.1.4).

The second new parameter of interest was the number of days that the space station violated the T^3 requirement. This statistic helped in comparing different strategies.

Furthermore, we gave the shuttle a possible delay of up to ten days. This delay consisted of a uniform distribution between zero and ten days.

Our altitude strategy reboosted the station to an altitude resulting in 360 days to 278 km without any onboard propellant.

4.3.3 Phase III

In the third phase, we altered the use of propellant and we added some more randomness to the model. In phase one we found that we were using all of the onboard propellant in an attempt to reboost as high as possible. In the second phase we only allowed the reboost to take us to the altitude that would result in 360 days to 278 km without any of the onboard propellant. This phase appeared successful, until we discovered an error in our assumption (see section 4.4.2). In the third phase, we fixed the erroneous assumption in phase two while restricting the liberal use of propellant from phase one. To accomplish this we allowed a reboost in phase three to use all of the fuel provided by the Progress M2, but did not permit onboard propellant use. We realize that this restricts the efficiency of the station and the tanks will continually, but slowly, build to capacity. Furthermore, this strategy does not take into account the upmass of the shuttle.

We also incorporated solar cycle prediction into our model. Up to now, the predicted decay used the *true* solar values for calculation. In the real prediction of decay, the true values are not available and must be predicted. To add some of this realism to our model we added a pseudo-prediction of solar activity. This prediction consisted of the mean solar cycle values created earlier, but did not contain the monthly variations found in our *true* values.

4.3.4 Phase IV

We arranged Phase IV to directly account for shuttle rendezvous and allow for use of onboard fuel. We planned for the station to reboost the day after the shuttle undocked, if possible. When not possible, the station reboosted early and tried to make the next shuttle. Although we allowed the use of all onboard propellant, we avoided the situation in Phase I. The situation in Phase I used almost all of the propellant and reboosted the station high into the atmosphere to attain T^3. This situation was avoided because the goal of Phase IV was to bring the station down for the shuttle.

4.4 Results

4.4.1 Phase I Results

The completion of phase one closed the loop of data and allowed us to observe how the space station fared with this strategy over the lifetime and subjected to various runs. By plotting the altitude of the space station over time, we found that this strategy

did not handle high solar activity well. Figure 4.7 demonstrates the problem we encountered with Phase I:

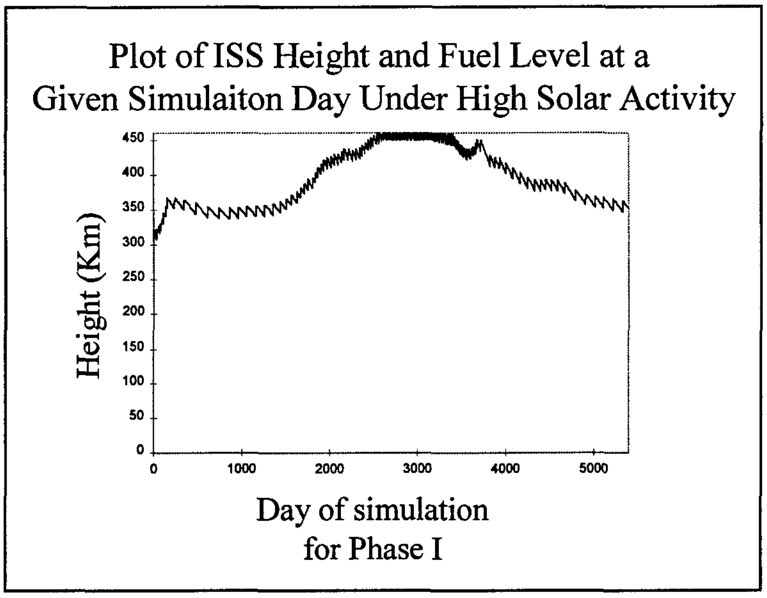


FIGURE 4.7 Plot of ISS Altitude from Phase I

From Figure 4.7 we observed the many reboosts conducted at the height of the solar cycle peak. Reboosts occurred every twenty days up to the maximum allowable altitude of 460 km (Russian design constraint). Because the reboost did not use all of the propellant, the remaining propellant went overboard and the amount of waste propellant reached ludicrous amounts. As the number of reboosts went up, the number of days of microgravity decreased until we observed some years where zero useable days of microgravity occurred.

4.4.2 Phase II Results

Phase two resulted in improved days of microgravity and number of reboosts. By constraining the number of days between reboosts to be at least sixty days, we

simultaneously limited the number of reboosts to a maximum of six per year. We also altered the reboost strategy. Instead of allowing a reboost using the full capacity of propellant, we reboosted the station to an altitude resulting in 360 days to 278 km without any onboard propellant. This strategy initially appeared to work extremely well. The minimum number of days of microgravity per year averaged 225 days with a variance of 16, as shown in Figure 4.8.

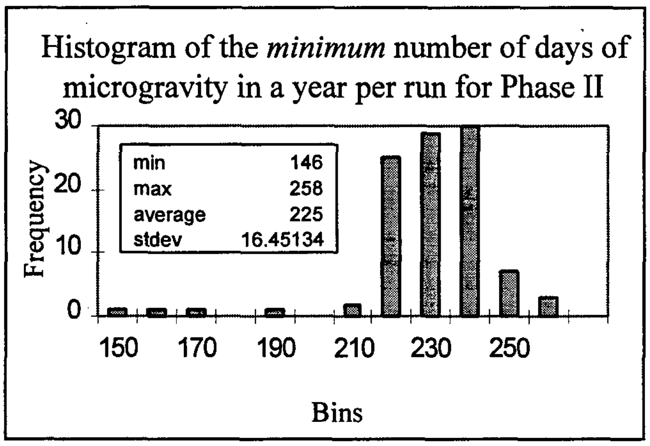


FIGURE 4.8 Phase II Microgravity Histogram

Figure 4.9 shows that the average maximum number of reboosts in a year for all runs in our model was 2.61. This value far exceeds the constraint of a maximum of six Progress flights per year and a desired four per year.

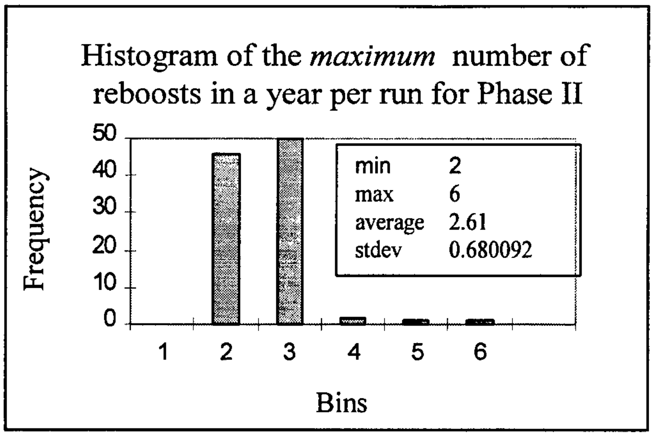


FIGURE 4.9 Phase II Reboost Histogram

Other parameters demonstrated similar good results. The maximum number of days below the lifetime altitude T^3 were very low, except for under extreme solar activity. This strategy also appeared to accommodate the shuttle dockings well. However, by observing the ISS altitude plot, we noticed a general trend influencing the entire lifetime of the station. Our assumption that the initial height of the station ranged from 350 to 420 overpowered the entire strategy. This altitude permitted the station to initially decay for **over three years** before the first reboost, as observed in Figure 4.10:

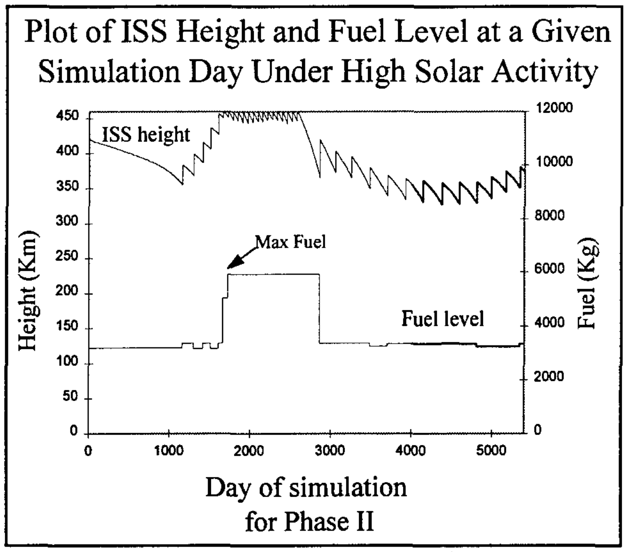


FIGURE 4.10 Phase II ISS Altitude Plot

We observed this long decay under all types of solar activity. When we altered the code to initialize the height with a similar strategy the rest of the profile followed, the station ended up crashing under periods of high solar activity. We then realized that we needed to improve the strategy to account for the initial altitude and the first few years of low solar activity. Phase three attempts a strategy that incorporates the same logic for the initial height as it does for reboosts.

4.4.3 Phase III Results

Phase three corrected the error uncovered in phase two and added a prediction capability for future decay. The reboost strategy in phase three required the reboost to use as much of the propellant from the Progress M2 as possible and did not authorize the use of onboard reserve propellant. This strategy fixed the initial height to correspond with the rest of the strategy and resulted in realistic looking altitude profiles, nice

parameter outputs, and histograms very similar to phase two. However, because the iteration scheme used for the calculation of propellant and reboosting does not use all of the propellant from the Progress M2, the onboard propellant slowly increases with each reboost. Once the tanks hit their capacity, all following reboosts will throw the remaining propellant overboard, as Figure 4.11 shows.

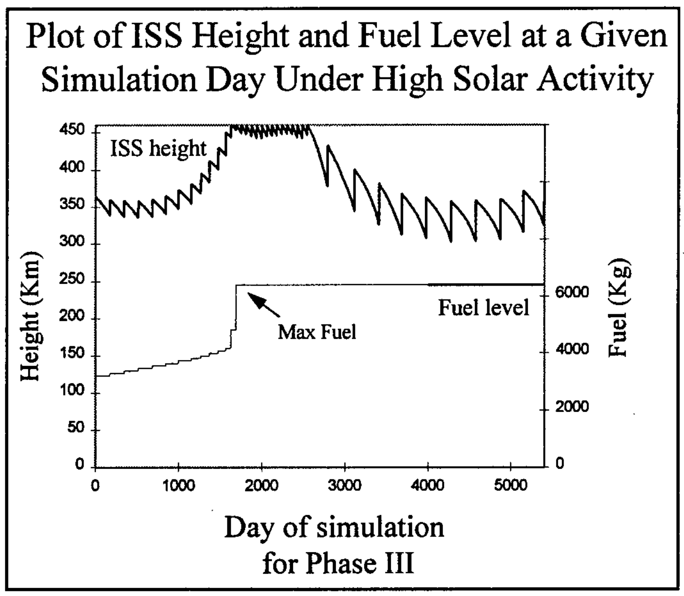


Figure 4.11 Phase III ISS Altitude Plot

Besides poor onboard fuel management, this strategy did not attempt to optimize the shuttle upmass.

However, in addition to correcting the initial height problem, this strategy added some uncertainty into the decay prediction (see section 4.3.3). Although this prediction uses the underlying form of the values used for the *true* solar activity, it does allow some

degree of randomness into the prediction. Future phases could incorporate the past, known, values of solar activity and project future short range values for decay. This phase gave us our first valid strategy. With a valid strategy that works under various ranges of solar activity, we then proceeded to vary our strategies to compare results and improve our parameters of interest. The goal of Phase IV was to directly plan for shuttle missions and dock lower in the altitude so that shuttle cargo upmass could increase.

4.4.4 Phase IV Results

Phase IV directly accounted for the shuttle missions and attempted to rendezvous the shuttle and the station in a manner that the station reboosted the day after the shuttle undocked. Due to some altitude limitations, this ideal docking could not always be attained and the station reboosted earlier than preferred. This phase also allowed onboard fuel use. The altitude profile shown in Figure 4.12 instantly shows the amount of time the station spends below the altitude of 358 km and therefore allowing negative penalties, or increases in cargo upmass.

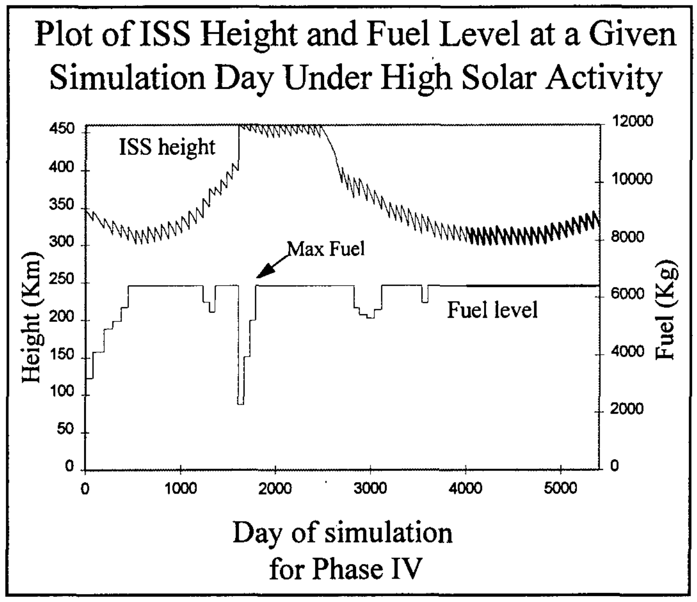


Figure 4.12 Phase IV ISS Altitude Plot

The fuel level, without looking at the altitude profile, piqued curiosity with the large dip. This large dip occurred because the station entered a drastic peak in solar activity and the station boosted about 70 km to react to this event. The onboard quickly built back up as the station remained at the top of the allowable altitude for the duration of the maximum solar activity. Comparisons between Phase III and Phase IV display the tradeoffs in parameter results for the different strategies.

4.4.5 Phase III and Phase IV Comparison

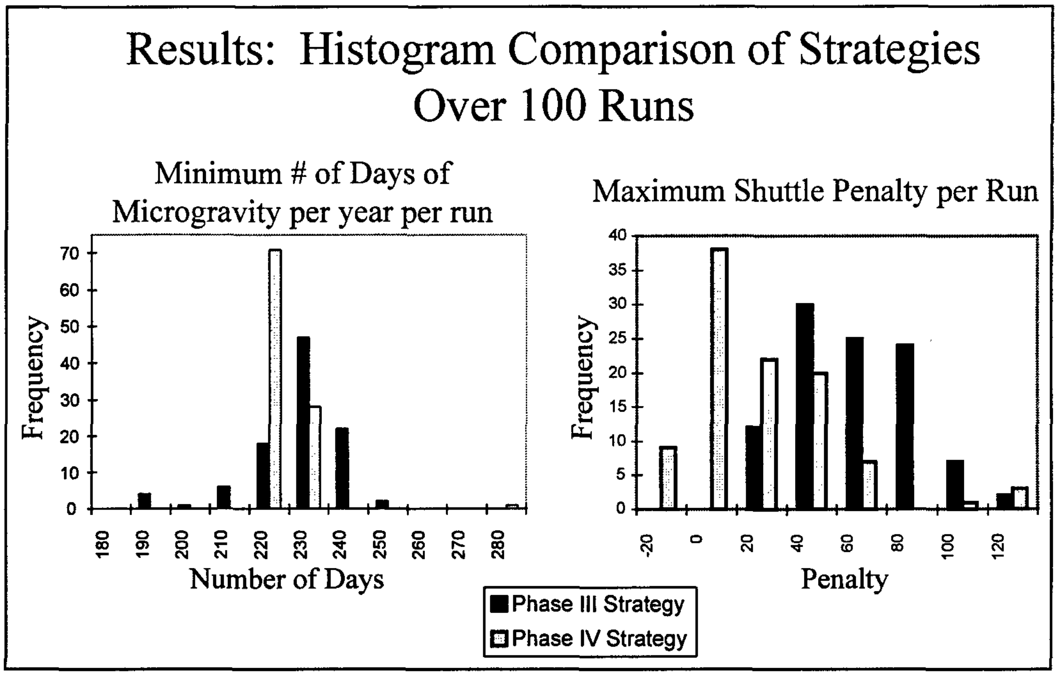


Figure 4.13 Comparison of Two Strategies

Figure 4.13 summarizes two key results from the two phases. As expected, when we brought the station lower in the atmosphere to allow larger shuttle upmass capabilities, the average number of days of microgravity per year per run decreased some compared to Phase III. However, a nice surprise resulted with a smaller variance in Phase IV than in Phase III, as demonstrated by the narrow range of Phase IV. Another expected result was the shift in shuttle penalties in the new strategy. Phase IV incurred less shuttle penalties than Phase III, thus verifying the new strategy accomplished the intended goal.

4.5 CONCLUSION

The completed phases in this study emphasize the many variations in the development of a dynamic strategy and reveal how randomness affects each strategy. We demonstrated how simple simulation models can look at the ISS operational planning in a new way which incorporates this random aspect and helps us re-strategize our approach. We also showed how various strategies can be easily compared using simple histogram charts. Chapter 5 will propose improvements to these models as well as future research possibilities.

5. Summary, Recommendations for Future Research, and Conclusions

5.1 Chapter Overview

This chapter brings this thesis research to an end by summarizing the purpose of the research and describing the importance of the results we obtained. In addition, it presents options for improving the models we created as well as provides overviews for potential helpful topics under the category of Operations Research. A conclusion terminates this research by summarizing advantages and disadvantages of our effort.

5.2 Summary and Significance of Results

The purpose of this thesis effort was to provide NASA with a stochastic perspective and some insights to the operational planning of the International Space Station (ISS). We looked at key issues involved in orbit and traffic planning and then created prototype models to represent operational issues. Our main focus in these prototype models was to directly account for random variations firmly embedded in the situation. We created a method of obtaining robust altitude strategies which could take advantage of low solar activity but which would hold up under high solar activity as well.

From the results we found that simulation modeling helped us to identify errors in our approach as well as missing strategies. It also allowed us to incorporate various solar cycle possibilities into our strategy and then test our strategies against the candidate, but unknown future cycles. By directly accounting for randomness, we allow for variations and deviations that may not otherwise surface.

5.3 Recommendations for Future Research

Our recommendations for future research are divided into two sections. The first section relates directly to improving the prototype models we developed. These suggestions expand the models by reducing the assumptions and simplifications. The second section proposes additional Operations Research tools to use in conjunction with simulation. These tools used simultaneously can provide even more insights and help improve decision making.

5.3.1 Enhancements for the Prototype Models

While this research allows us to evaluate various strategies while taking into account random behaviors in the system and simplifying assumptions that prevent direct operational use of this research. To improve these models, details need to be added to the models. Further additions could allow actual solar activity to be updated and reflected in the predicted solar activity. While the Shuttle flights included a random dimension we did not model cancellations. Neither did we model slips and cancellations in either of the Soyuz or Progress flights. We also did not plan for maintenance of small adjustments in attitude control. In addition, we assumed reboosts were instantaneous versus a real reboost time frame. Another possibility for additional iterations is to allow for the rescheduling of the remaining (non-Progress) flights, including slips and cancellations. Others could explicitly plan for the microgravity blocks requirement. Finally, iterations need to project logistical requirements for flights.

5.3.2 Possible Operations Research Techniques to Incorporate

5.3.2.1 Optimization

Optimization seeks to allocate limited resources to certain activities in a way that best satisfies an objective.

Linear programming describes a problem using a mathematical model. The model seeks to either maximize or minimize a mathematically formulated objective function subject to similarly mathematically formulated constraints. The following is the general formulation for a minimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) \le 0 \\ & h(x) \le 0 \\ & x \ge 0 \end{array}$$

For example, if we wanted to minimize the cost erecting a wire and wooden fence subject to the amount of materials available, we would set up a math model to solve this problem. The model would look something like the following:

minimize cost = 3(length of wire) + 5(length of wood)

subject to: length of wire $\leq$ 15 feet

length of wire $\geq$ 0 feet

length of wood $\leq$ 30 feet

length of wood $\geq$ 0 feet

length of wood + length of wire = 20

Solving this model would result in 15 feet of wire and 5 feet of wood for a total cost of 60 dollars. This solution would be the best, or optimal solution. Any other solution would either violate the given conditions or would cost more that 60 dollars.

One application of linear programming would be to maximize the amount of cargo to carry up to the station subject to constraints such as capacity and essential amounts of certain items.

In a linear programming model, the mathematical functions in the model are required to be linear functions. Nonlinear programming relaxes the requirement that the functions need to be linear.

Nonlinear programming takes into account the fact that not all behaviors are linear in nature. Two examples of non-linear functions are decay rates and solar activity. Sometimes these functions can be reformulated into a linear programming format. Other times special algorithms and techniques are needed to solve these problems.

Integer linear programming, or integer programming, is a special type of programming in that all of the solutions must have integer values. For example, we cannot send up a fraction of a shuttle. Integer programming often uses various techniques like either-or constraints to enforce integer solutions in linear programming models. Unfortunately, constructing a linear programming model to enforce integer solutions quickly can become computationally infeasible. Different techniques such as branch-and-bound and cutting-plane algorithms attempt to reduce the possible solutions to be examined for optimality.

Specific integer programming problems such as the transportation problem or the assignment problem constitute the basis of scheduling problems. The next section briefly describes the subject of scheduling problems.

5.3.2.2 Scheduling

Scheduling is a form of decision-making that attempts to allocate limited resources to tasks over time with the goal of optimizing one or more objectives (Pinedo, 1995: xiii,1). Resources may take such forms as amount of propellant, altitude above the earth, capacity, logistics, or days of microgravity. Tasks may include reboosts, docking, or loading vehicles. Various objectives could be to optimize the upmass of the space shuttle or the amount of propellant used for a rendezvous for the space station and the space shuttle. *Heuristics* are rules of thumb that schedulers often use because the size of scheduling problems easily become so large that they are unsolvable in a reasonable time frame. For example, a heuristic could be to always schedule a space shuttle rendezvous ten days before a reboost occurs.

Scheduling with time windows is a specific type of scheduling problem that allows certain events to only occur during specific periods of time. One way of solving this problem is to assign weights to different events and their time of occurrence and try to maximize the sum of the weights times the events. For example, we attached penalties to the altitude at which the space shuttle docked. If they docked at an altitude above 358 km, they received a penalty corresponding to loss of upmass. To plan for shuttle dockings, an scheduling optimization algorithm or heuristic could be developed to

minimize the sum of the shuttle penalties. Another similar OR technique is Inventory Modeling.

5.3.2.3 Inventory Modeling

Inventory modeling aids decision makers in resolving questions such as when to order and how much to order. Figure 5.1 is a diagram of a typical inventory level plot. This plot shows how the level of goods decreases with a constant rate of demand. When the inventory level reaches a pre-determined level, the decision maker reorders the commodity so that the level can be replenished when it reaches zero.

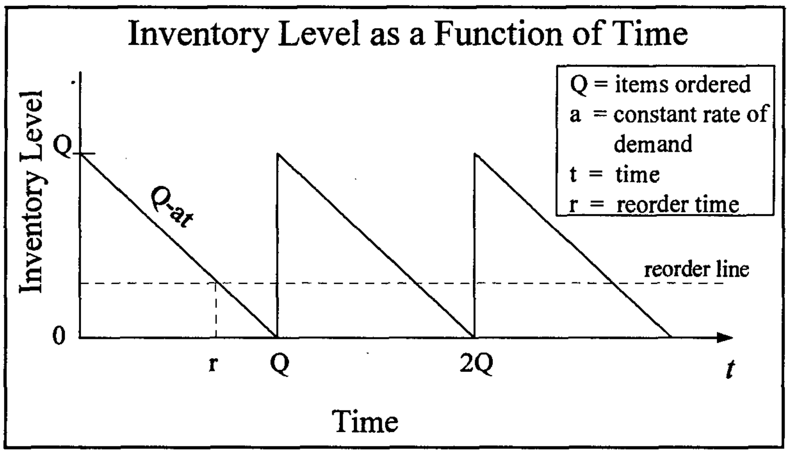


FIGURE 5.1 Inventory Diagram (Hillier, 1990: 692)

This diagram is a basic representation of inventory modeling. Other factors that could be included are whether to allow shortages or extra inventory or if the demand is not a constant rate.

If the demand is not a constant rate, then the inventory level must be reviewed in order to determine when the level dips below the reorder line. One way to do this would

be to continually review the level of inventory. A more efficient way, however, would be to review the level of inventory periodically. If the level of inventory is below the reorder point, then an order is released. Figure 5.2 demonstrates a non-constant demand rate and intervals of periodic review.

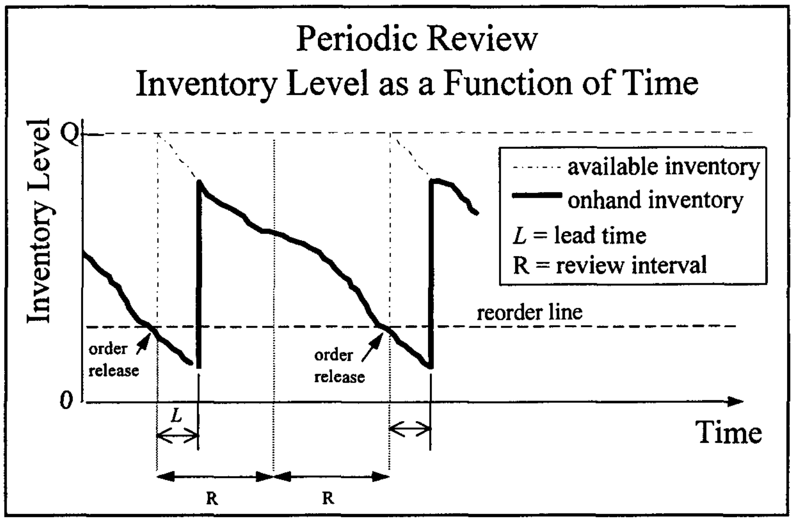


Figure 5.2 Periodic Review Inventory Diagram (Hax, 1984: 225)

Looking at the inventory diagram, one can see that it closely resembles an altitude profile. One application of inventory modeling would be to set altitude as a commodity and decay as the demand over time. When the station reaches the "reorder line," the station is reboosted. The reorder line could be a constant altitude, or a lifetime altitude such as our T^3. For the periodic review, it may not be wise to have constant review intervals but to review more often as the station approaches the *reorder* line.

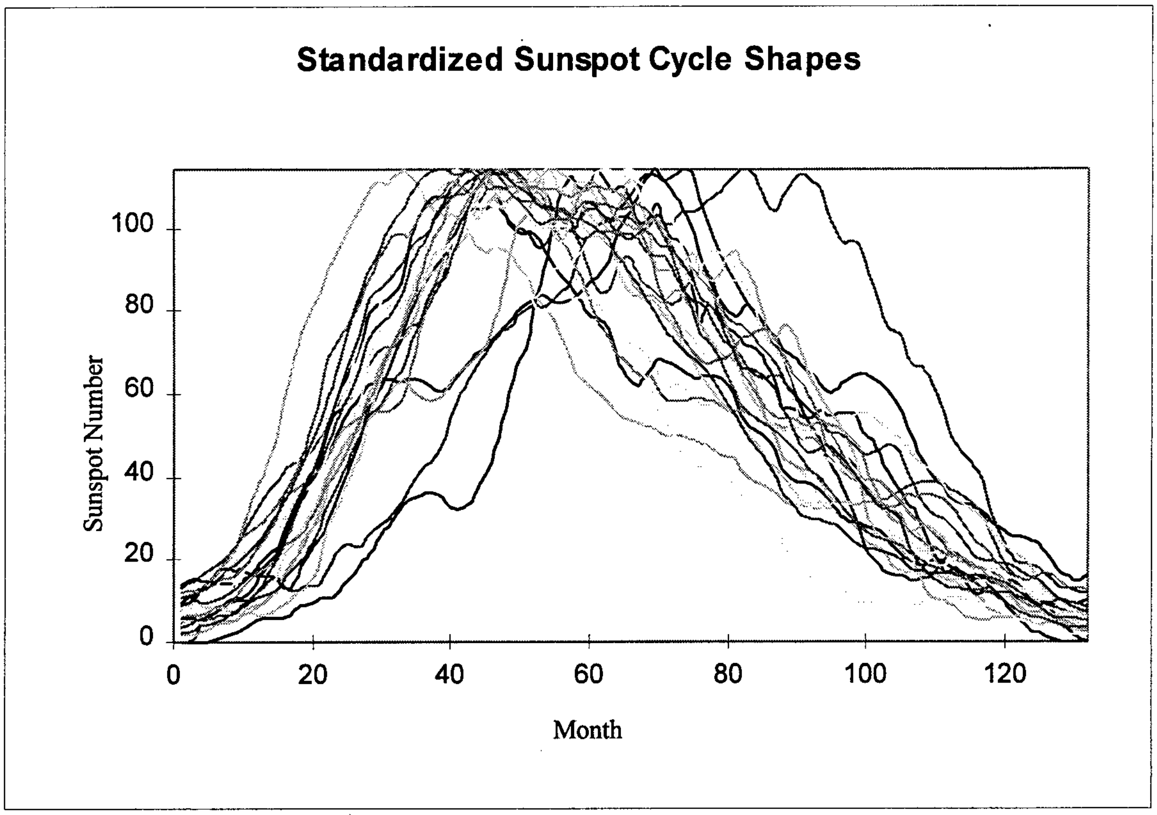
5.4 Conclusions

Although this research effort does not terminate with a final product or tool to hand over for implementation, it does provide NASA with a format for stochastic modeling of the space station operations that can benefit their planning process. Simulation modeling is a tool that can be used to incorporate the natural random behaviors which affect the lifetime of objects in low-earth-orbit. We used simulation to create prototype models of their planning process to analyze current altitude strategy approaches and acquire new strategies from insights observed. In addition, by extrapolating random future solar activity values from the interpolation of historical data , we established a spectrum of possible solar activity rather than just maximum, mean, and minimum values. While we know that none of these future solar cycles will be an exact representation of the next, behaviors closely reflecting the future cycle should emerge from a distribution of these runs. From this procedure, we demonstrated how we can analyze a strategy using distributions of parameter outputs in response to random inputs. Finally, we set a foundation for future research that could culminate in a final product to exercise in the operational planning process.

Appendix A: Length FORTRAN Program

```
*********************************************************************************************
*   TITLE: Length Program (length.for)
*  AUTHOR: 2d Lt Jillene B. Rylaarsdam and Major E. Price Smith
*  DATE: March  1996
*  DESCRIPTION:  This program reads in the actual solar cycle data and puts it in uniform length.  This
*     data is  read into an excel file to put into uniform height.
*   FILES USED:
*   Input file: spot3.txt
*   Output files:  spot3.out    spot3.dat
*    VARIABLES:
* Main Program:
*  i,j,k = integer counter variables
*  length = integer array for the length of the historical solar cycles
*  spots = real array for the historical data in the historical solar cycles
*  shape = real array for the historical data standardized in length
*************************************************************
      integer i,j,k,length(22)
      real spots(22,164),shape(22,132)
      open(unit=1,file='spot3.txt',status='old')
      open(unit=2,file='spot3.out',status='unknown')
      open(unit=10,file='spot3.dat',status='unknown')
* Read in historical data
      do i=1,22
       read(1,*)length(i),(spots(i,j),j=1,length(i))
       print*,length(i)
      end do
* Tie down the endpoints and then standardize the datato a length of 131
      do i=1,22
       shape(i,1)=spots(i,1)
       shape(i,132)=spots(i,length(i))
       do j=2,131
        do k=1,length(i)-1
         if(j.ge.(k*132./length(i)).and.
     &        j.le.((k+1)*132./length(i)))then
          shape(i,j)=spots(i,k) +
     &              (spots(i,k+1)-spots(i,k)) *
     &                (j-k*132./length(i)) /
     &                ((k+1)*132./length(i)-
     &                 k*132./length(i))
         end if
        end do
       end do
       write(2,*)(shape(i,j),j=1,132)
      end do
* Write standardized lengths to a file
      do j=1,132
       write(10,200) j,(shape(i,j),i=1,22)
200     format(1x,i3,22(f8.3))
      enddo
      end
```

Appendix B: Plot of Standardized Sunspot Cycle Shapes



Appendix C: Solar Model FORTRAN Program

```
************************************************************************
*
*      TITLE: Solar Model Fortran Program  (solarmod.for)
*     AUTHOR: Major E. Price Smith and 2d Lt Jillene B. Rylaarsdam
*       DATE: March 1996
* DESCRIPTION:     This model reads in the standardized solar cycle data and randomly
*          chooses 2 streams. The two streams are combined in an almost convex
*          combination.  The stream is then re-standardized to the common
*          height.  From the standard height and length, the program will draw
*          random numbers to vary the height and the length of the stream.
*          Monthly deviations are then randomly chosen.  The final stream is one
*          that has a shape of two historical streams, but which varies in height,
*          length, and monthly deviations.
*             Three of these streams will then be augmented to form three
*          continuous cycles.  For the first cycle, a random variate of 12-36
*          months will be chosen to draw the "tail" of the cycle.  The second
*          cycle will be used in its entirety.  As we are simulating 15 years (or
*          1880 months) of lifetime, plus one year for advanced decay calculations
*          in the decay model, the length of the third cycle will be:
*            195 months - (length of first cycle's tail + length of second cycle)
*          This loop will give us the solar activity for one future solar world.
*             The process will then be repeated 100 times to show the
*          different worlds which can occur.  This data will be placed in arrays
*          which will then be printed to files.  Two files will be used for input
*          into Excel and two other files will be used for input into the Decay/
*          Reboost Model.  The prediction files will be used from Phase 3 on to
*          "predict" the future values.  These are just the random length and height
*            values before the monthly random variation was added.
*  FILES USED:
*  Input file:  standsun.in       solar cycle shapes which have been standardized
*                          to common lengths and heights
* Output files: mixstreams.out    follows the progression of manipulating streams
*            supercyc.out     outputs in a format to be used by Excel
*            supercyc.in      outputs in a format to be read by Decay/Reboost Model
*            predcyc.out      outputs in a format to be used by Excel (for Phase 3 on)
*            predcyc.in       outputs in a format to be read by Decay/Reboost Model
*                             (Phase 3 on)
* SUBROUTINES:
*            none
*   VARIABLES:
* Main Program:
*   a = real*8 value - low value of the TRIANG distribution
*   b = real*8 value - value where the peak occurs in a TRIANG distribution
*   bottom = integer used for writing to files, if the length of a cycle is greater
*          than the standard length, the new length is the bottom
*   c = real*8 value - high value of the TRIANG distribution
*   cyc =   integer for looping.  Computes new cycle stream three times to
*          augment the cycles.
*   cyc1(3,1000) = real*8 array.  Stores the values of each of the three cycle streams
*   cyc2(3,1000) = real*8 array.  Stores the "predicted" values of each of the three cycle streams
*   dev = real*8 IID Normal(0,1) clipped variable (-3,3) used in determining the monthly
```

```
*          deviation.  "Z-statistic"
*     devht = real*8 IID Normal(0,1) clipped variable (-3,3) used in determining the height
*          deviation.  "Z-statistic"
*     devlg = real*8 IID Normal(0,1) clipped variable (-3,3) used in determining the length
*          deviation   "Z-statistic"
*     drand = real*8 intrinsic function variable used in computing random, uniform(0,1)
*     hite =  real*8 variable for height of cycle.   hite = devht(stdev) + mean
*     i   =  integer counter variable
*     i100th = real*8 variable that retrieves the 100th of a second from the time
*     ihr = real*8 variable that retrieves the hour from the time
*     imin = real*8 variable that retrieves the minute from the time
*     isec = real*8 variable that retrieves the second from the time
*     iseed = real*8 function variable that combines the time variables to find a random
*          seed
*     j = integer counter variable
*     k = integer counter variable
*     legcyc(3) = integer array which holds the partial lengths of the three cycles
*          legcyc(1) = length of tail of cycle one (randomly between 12-36 months)
*          legcyc(2) = length of cycle two
*          legcyc(3) = length of beginning of cycle three
*                    = 195 months - (legcyc(1)-legcyc(2))
*     leng(3) = integer array which holds the full lengths of the three cycles
*     length = real*8 variable for length of cycle. length = devlg(stdev) + mean
*     lots1(3,1000) = real*8 array.  Stores the values of each of the three cycle streams
*          three-cycle worlds computed.  The second is the month of the augmented
*          cycle.  0: the length of the three-cycle world (in months).
*     lots2(3,1000) = real*8 array.  Stores the value of the predicted three cycle streams
*     manycyc = integer variable for the number of three-cycle worlds
*     maxht = real*8 variable to find the maximum height of a cycle in order to
*          re-standardize height of the combined streams
*     mix_stream(0:4,1000) = real*8 array which contains the various manipulations
*          of the combined stream:
*         !0:month,1:unstretch,2:amplified,3:stretch,4:deviated
*     num_points = integer variable - number of points in solar data per cycle (=132)
*     num_stream = integer variable - number of streams to use in mix  (MAX 22)
*     peak = real*8 value - x-value where the peak height of the Triangular distribution occurs
*     r = real random uniform(0,1) variable
*     seed = real*8 predetermined seed set by programmer used for random number generation
*     sflux(0:22,1000) = real*8 array reads in the historical sunspot data and then is
*          converted to solar flux data.
*     sfs(22) = integer array used to determine which solar streams are used in mixing
*     sum = real*8 variable used to sum up the total weights of the streams
*     totleng = integer variable - number of months for simulation (currently=15 years)
*     u1 = real*8 random, uniform (0,1) variable used to generate Normal(0,1) variate
*     u2 = real*8 random, uniform (0,1) variable used to generate Normal(0,1) variate
*     v1 = real*8 variable used to generate Normal(0,1) variate  v1 = 2*u1 - 1
*     v2 = real*8 variable used to generate Normal(0,1) variate  v2 = 2*u2 - 1
*     w = real*8 variable used to generate Normal(0,1) variate   w  = (v1**2) + (v2**2)
*        (w is less than or equal to 1)
*     worlds = integer variable for the number of worlds to create
*     wt(22) = real*8 array which holds the weights of the streams used in mixing
*     y = real*8 variable used to generate Normal(0,1) variate   y = sqrt((-2*log(w))/w)
*********************************************************************************************
*********************************************************************************************
```

```
*
*              SOLAR MODEL
*
***************************************************************************************
     program solar
     implicit none
*////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR SOLAR MODEL
     integer pworlds,months
     parameter (pworlds=101)
     parameter (months=195)
     real*8 sflux(0:22,195),wt(22),u1,u2
     real*8 v1,v2,w,y,dev,sum,peak
     real*8 hite,length,devht,devlg,ddev,maxht,a,b,c
     real*8 mix_stream(0:4,months)
          !0:month,1:unstretch,2:amplified,3:stretch,4:deviated
     integer i,j,k,num_points,sfs(22),num_stream, bottom,cyc,worlds
     integer legcyc(3)
     real*8 cyc1(3,months), cyc2(3,months),lots1(pworlds,0:500)
     real*8 lots2(pworlds,0:500)
     integer totleng,leng(3),manycyc
     real*8 r, drand
     integer seed
*--- INITIALIZE VARIABLES FOR SOLAR MODEL
     worlds = pworlds
     num_points=132    !number of points in solar data per cycle
     totleng = months   !number of months for simulation (15 years) plus
                   ! one year for advance decay calculations in
                   ! decay model
*////////////////////////////////////////////////
*/ OPEN FILES FOR DECAY/REBOOST MODEL
*////////////////////////////////////////////////
*  Files for Solar Model
      open(unit=11,file='standsun.in',status='old')
      open(unit=12,file='mixstrms.out',status='unknown')
      open(unit=13,file='supercyc.out',status='unknown')
      open(unit=14,file='supercyc.in',status='unknown')
      open(unit=15,file='predcyc.out',status='unknown')
      open(unit=16,file='predcyc.in',status='unknown')
*--- INPUT DESCRIPTIVE HEADER FOR FILE
      write(12,*) 'deviations   world   height   length   daily'
*////////////////////////////////////////////////
*/  READ SOLAR NUMBER HISTORIES
*////////////////////////////////////////////////
*  NOTE: sflux(0,i)=i always
      do j=1,num_points  !num_points=132=13years*12 mo/yr
       read(11,*)sflux(0,j),(sflux(i,j),i=1,22) !22 solar cycles
      end do
*////////////////////////////////////////////////
*/ INITIALIZE RANDOM UNIFORM(0,1) GENERATOR
*////////////////////////////////////////////////
* Initialize random number generator and then call uniform(0,1) random
```

```
* variable r=drand(0)
      seed = 7651234
      r = drand(seed)
* Determine number of streams to mix
      r=drand(0)
      num_stream= 2!  1+int(5*r) !number of streams to use in mix  (MAX 22)
*////////////////////////////////////////////////
*/ BEGIN LOOP TO CREATE MANY DIFFERENT CYCLES
*/ MEAT OF THE PROGRAM
*////////////////////////////////////////////////
    do manycyc=1,worlds
* Three cycles for tail end of one, one full cycle, and beginning of one
     do cyc=1,3
*/////////////////////////////////////////////////////
*/ BEGIN TO MIX HISTORICAL CYCLES AND CREATE "NEW" CYCLES
*/////////////////////////////////////////////////////
*---DETERMINE WHICH SOLAR FLUX STREAMS TO MIX
      do i = 1,num_stream
       r=drand(0)
       sfs(i) = 1+int(22*r)
      enddo
*--- DETERMINE WEIGHTS FOR EACH STREAM TO MIX
      sum = 0
      do i = 1,num_stream,2
       r=drand(0)
       wt(i) = 1.047*r
       sum = sum+wt(i)
       if (i.lt.num_stream) then
        wt(i+1) = 1.- wt(i)       !pseudo-convex combination
        sum = sum+wt(i+1)
       endif
      enddo
      do i=1,num_stream
       wt(i) = wt(i)/sum
      enddo
*--- MIX STREAMS
      do i=1,num_points !=132
       mix_stream(0,i) = i
       mix_stream(1,i) = 0 !initialize to 0 before averaging
       do j = 1,num_stream
        mix_stream(1,i)=mix_stream(1,i)+wt(j)*sflux(sfs(j),i)
       enddo
      end do
*--- RE-STANDARDIZE HEIGHT TO 114
*    Find max height
      maxht = 0
      do i=1,num_points
       if(mix_stream(1,i).gt.maxht) then
        maxht = mix_stream(1,i)
       endif
      enddo
      do i=1,num_points
       mix_stream(1,i)=mix_stream(1,i)*114/maxht
      enddo
```

```
*--- STRETCH HEIGHT
*    compute a triang(40,77,205) cycle HEIGHT
      a = 40  !low value
      b = 77  ! value where peak occurs
      c = 205 ! high value
      r=drand(0)
      u1 = r
      peak = (b-a)/(c-a)
      if (u1.le.peak) then
       devht = sqrt(peak*u1)
      else
       devht = 1 - sqrt((1-peak)*(1-u1))
      endif
      hite = a+devht*(c-a)
      do i=1,num_points
       mix_stream(2,i)=mix_stream(1,i)*(hite)/114
      end do
*--- STRETCH LENGTH
*    compute a Triang(105,119,172) cycle LENGTH
      a = 105  !low value
      b = 119  ! value where peak occurs
      c = 172 ! high value
      r=drand(0)
      u1 = r
      peak = (b-a)/(c-a)
      if (u1.le.peak) then
        devlg = sqrt(peak*u1)
      else
        devlg = 1 - sqrt((1-peak)*(1-u1))
      endif
      length = a + (c-a)*devlg
      length = int(.5+length)

* tie down the endpoints of the stretched cycle
      mix_stream(3,1)=mix_stream(2,1)
      mix_stream(3,length)=mix_stream(2,num_points)
      do j=2,length-1  !j is month of new stretched cyle
       do k=1,131     !k is the month of mixed and amplified standard cycles
        if(j.ge.(k*length/132.).and.
   &       j.le.(((k+1)*length)/132.))then
          mix_stream(3,j)=mix_stream(2,k) +
   &              (mix_stream(2,k+1)-mix_stream(2,k)) *
   &                (j-k*length/132.) /
   &                ((k+1)*length/132.-k*length/132.)
        end if
       end do
      enddo
*--- RESIZE ALL DATA POINTS
      do i=1,length
*--- MAKE AN N(0,1) DEVIATE FOR MONTHLY DEVIATIONS
*    (from Law and Kelton, Simulation Modeling & Analysis page 491)
35      r=drand(0)
       u1=r
       r=drand(0)
```

```
          u2=r
          v1 = 2*u1 - 1
          v2 = 2*u2 - 1
          w  = (v1**2) + (v2**2)
          if (w.gt.1) then
             goto 35
          else
           y = sqrt((-2*log(w))/w)
           dev = v1*y      ! an IID N(0,1) random variate
           if (dev.gt.3.or.dev.lt.-3) goto 35
           ddev = dev*5
          end if
          write(2,175) dev
175         format('u (-3,3) = ',f10.3)
          mix_stream(4,i)=mix_stream(3,i)+(ddev)     ! deviate month
          if (mix_stream(4,i).lt.0) then
            mix_stream(4,i) = 0
          endif
         end do
*--- WRITE STRETCHED, AMPLIFIED, AND DEVIATED DATA TO FILE 'MIXSTRMS.OUT'
         do i=1,length
           cyc1(cyc,i) = mix_stream(4,i)
           cyc2(cyc,i) = mix_stream(3,i)
         enddo
         leng(cyc) = int(length)
         write(12,25) worlds,devht,devlg,dev
25        format(14x,i5,5x,f10.5,5x,f10.5,5x,f10.5)
         if (length.gt.132) then
          bottom = int(length)
         else
          bottom = 132
         endif
         do j=1,bottom
          write(12,125) j,(mix_stream(k,j),k=1,4)
125         format(1x,i4,3x,4(3x,f10.6))
         end do
        enddo
*--- Determine length of tail of first cycle (uniform 1-3 years)
        r=drand(0)
        legcyc(1) = (1+(2*r))*12
        legcyc(2) = leng(2)
        legcyc(3) = 194 - (legcyc(1)+legcyc(2))
        k = 0
*--- TAKE THREE CYCLES AND AUGMENT TO ONE LONG STREAM
        do cyc=1,3
         if (cyc.eq.1) then
          do i=(leng(1)-legcyc(1)),leng(1) ! Tail of the first cycle
            k = k+1
            lots1(manycyc,k)=cyc1(cyc,i)   ! Place the tail of the first cycle on a
            lots2(manycyc,k)=cyc2(cyc,i)   ! stream which will contain all three cycles,
                                           ! augmented.
          enddo
         else
          do i = 1,legcyc(cyc)             ! Full second and partial third cycle
```

```
        k = k+1
        lots1(manycyc,k)=cyc1(cyc,i)   ! Augment the second and third cycles
        lots2(manycyc,k)=cyc2(cyc,i)   ! to the first cycle in one long stream
       enddo
      endif
     enddo
    enddo
*--- RECORD THE TOTAL LENGTH OF EACH WORLD
    do i=1,worlds
     lots1(i,0) = totleng
     lots2(i,0) = totleng
    enddo
*///////////////////////////////////////////////
*/ BEGIN WRITING INFORMATION TO FILES
*///////////////////////////////////////////////
*--- WRITE TO FILE TO USE IN EXCEL GRAPHS 'supercyc.out'
    do k=0,totleng
     write(13,700) (lots1(i,k),i=1,worlds)
     write(15,700) (lots2(i,k),i=1,worlds)
700   format(2x,(100(f10.3,2x)))
    enddo
*--- WRITE TO FILE TO USE IN 'DECAY/REBOOST' MODEL 'supercyc.in'
    do i=1,100
     do k=1,totleng
      write(14,800) lots1(i,k)
      write(16,800) lots2(i,k)
800   format(f10.5)
     enddo
    enddo
*///////////////////////////////////////////////
*/ CLOSE FILES
*///////////////////////////////////////////////
    close(11)
    close(12)
    close(13)
    close(14)
999  end
```

Appendix D: Decay/Reboost FORTRAN Program for Phase III

```
**********************************************************************
*      TITLE: Decay/Reboost Model Fortran Program (dmIII.for) Phase III
*     AUTHOR: 2d Lt Jillene B. Rylaarsdam and Major E. Price Smith
*       DATE: March 1996
* DESCRIPTION: This model reads in the different three-cycle worlds from a file and
*         stores the monthly solar data in an array. The data is read in as the
*         12-month smoothed Zurich sunspot number and then converted to the solar
*         flux value using NASA's equation from TM-82478. A loop allows many
*         three-cycle worlds to be run. The number of reboosts completed in a world
*         begins at zero, as does the "wasted" propellant. The minimum number of
*         days between a reboost is 60. The initial altitude begins at an altitude
*         that is 360 days to 278 km. The density at the altitude is calculated
*         daily and the T3 altitude is calculated every 5 days. The T3
*         altitude is defined as the altitude that the station is at after reboosting
*         and decaying for 360 days. If the altitude after 360 days of decay
*         is less than or equal to 278 km (360 days to 278 km is required by NASA)
*         and the 60 day between reboost constraint is not violated, the station will
*         be reboosted. A Progress M2 will be used for the reboost. The reboost
*         will be accomplished in an attempt to use all available Progress propellant. No
*         onboard may be used. An iteration scheme was used to determine the amount
*         of propellant to use, with a given tolerance between the desired and the actual.
*         If the station is reboosted to 460 km (Russian hardware design constraint)
*         the reboost will terminate and any remaining propellant will first transfer to
*         the station and, once full, the remaining waste propellant will be thrown
*         overboard. Waste propellant will be calculated through the lifetime. The
*         daily altitude, waste propellant, and number of boosts will be kept as
*         statistics.
*
* FILES USED:
* Input file: supercyc.in     "Actual" many three-cycle worlds to simulate
*             predcyc.in      Predicted many three-cycle worlds to simulate
* Output files: decayreb.dat  Keeps track of month,day,height,and density
*             height.dat      Keeps track of height and outputs in a format to be
*                             used by Excel
*             prop.dat        Keeps track of propellant data
*             decinfo.dat     Keeps track of initial altitude, number of reboosts,
*                             and waste per three-cycle world lifetime
*             dci__.dat Keeps track of altitude and fuel level for Excel
*             microg.dat      Calculates the number of days of micro-G per year
*                             (in blocks of 30 or more)
*             reboost.dat     Calculates the number of reboosts per year
*             shuttle.dat     Keeps track of shuttle penalties per shuttle
* Hierarchy of the program, subroutines and functions
*
*                  DECMOD
*                 /  |  \ \
*          predecay   TTT  | |
*                   /   \| |
*                   |  Reboost |
*                    \   |   /
*                    density
```

```
*  FUNCTIONS:
*        density       calculates the density given the altitude and level
*                      of solar activity
*  SUBROUTINES:
*        predecay      determines the altitude the ISS would end up at if
*                      no reboosts occurred for 360 days. If the altitude is
*                      less than 278 km, then it determines the altitude that we
*                      need to reboost to in order to get 360 days to 278 km.
*        reboost       given the altitude and amount of propellant available
*                      for use, reboosts the ISS using the available propellant
*        TTT           determines the altitude that the station needs to be at in
*                      order for the station to be at about 278 km in 360 days after
*                      reboosting the station with all available onboard propellant.
* VARIABLES:
* Main Program:
*   B = constant for the ballistic coefficient quantity = Cd*A/mass
*      Cd = constant for the coefficient of drag
*      A = presented area of the space station
*      m = mass of the space station
*   BC = ballistic coefficient w/out shuttle attached
*   BN = constant for the ballistic number = mass/(Cd*A)
*   boost = integer counter variable to caculate number of boost in 180 months
*   Bshutt = Ballistic coefficient w/ shuttle attached
*   calendar(world,day) = integer time array to get a calendar input
*      Events: 1 = Shuttle dock/undock  2 = Soyuz dock/undock
*              3 = Soyuz undock      4 = Progress undock
*   case =
*   d = integer counter variable for a loop
*   day  = integer counter variable for the number of days to simulate (30 per month)
*   dd = integer variable annotating if the Progress can undock (no shuttle attached)
*   dday = integer variable annotating the day the reboost occurs (1-5400)
*   dayseconds = real*8 variable to convert days to seconds
*   decay = real*8 variable annotating the altitude (in km) that the ISS would be at if
*        no reboosts occurred
*   dec_alt = real*8 variable set equal to 'decay' variable after calculating function
*   density = real*8 function variable (see above description)
*   drand = real*8 intrinsic function variable - outputs random, uniform(0,1) varaite
*   endprop = real*8 variable - stores the amount of propellant left after reboost
*   event(world,day) = indicates microgravity disturbances, used to calculate quiet periods
*   fake = logical variable to determine if we are doing a fake reboost or not
*      .true. = fake reboost   .false. = real reboost
*   fuel(world,day) = real*8 variable designating the current amount of propellant
*             in the storage tanks of the ISS
*   H = real*8 variable - height of ISS above the surface of the earth
*   height(world,day) = real*8 array - stores the heights of the ISS during its lifetime
*        for each of the three-cycle worlds calculated
*   help = logical variable to determine if we are out of propellant for a reboost
*   i = integer counter variable
*   i100th = real*8 variable that retrieves the 100th of a second from the time
*   ihr = real*8 variable that retrieves the hour from the time
*   imin = real*8 variable that retrieves the minute from the time
*   info(world,0:16) = real*8 array that keeps track of vital information per three-cycle
*        0 = initial altitude
*        1 thru 15 = number of reboosts years 1-15
```

```
*         16 = amount of wasted propellant per world lifetime
*    isec = real*8 variable that retrieves the second from the time
*    iseed = real*8 function variable - combines the time variables to find a random seed
*    m = integer counter variable
*    mgdays = integer counter for days of microgravity
*    mass = constant - mass of the ISS
*    microg(world,year) = integer array that keeps track of days of micro-gravity each year
*    minreb = integer counter - number of days since last reboost, updated daily (to make sure
*         that there are at least 20 days between reboosts)
*    month = integer counter variable for the number of months to simulate per cycle
*    mu = constant - gravitational parameter = Ge*M in (km)^3/(sec)^2
*       Ge = Universal gravitational constant = 6.67x10^(-11) in Nm^2/kg^2
*       M  = mass of the earth  (kg)
*    newalt = real*8 variable
*    newH = real*8 variable - new height of ISS after reboost
*    onprop - real*8 variable - amount of propellant onboard (kg)
*    p = real*8 variable - equals 'density' variable after calculating function
*    penalty(worlds,75) = real*8 array - penalty for docking other than 358 km
*         '+' = bad (got a penalty)  '-' = good (negative penalty)
*    pflux(world,0:195) = real*8 array reads in the "predicted" sunspot data and then is
*         converted to solar flux data.
*    progM2 = constant - available useable propellant from Progress M2  (kg)
*    pworlds = parameter - the number of runs we are simulating
*    r = real random uniform(0,1) variable
*    Re = constant - radius of the earth (km)
*    rebalt = real*8 variable - lower altitude from which to reboost in (km)
*    restricted(world,day) = integer array that does not allow either Progress or Soyuz
*                  to dock or undock while the shuttle is attached
*    seed = real*8 predetermined seed set by programmer used for random number generation
*    sshut = integer value for the shuttle number = 1-75  (5 per world)
*    shuttle = 0-1 variable to determine if shuttle is attached
*          0 = shuttle is not attached    1 = shuttle is attached
*    sflux(world,0:195) = real*8 array reads in the "actual" sunspot data and then is
*         converted to solar flux data.
*    skipper = integer variable to determine if the days between reboosts is at least 60
*    slip = real*8 variable - equals +(0-10) days that the shuttle takes off
*    solar_flux = real*8 variable equal to sflux(world,month) for each loop
*    sshut = integer variable - index for the day the shuttle undocks
*    T3alt = altitude of station to make 278 km in 360 days w/ reboost using all prop
*    tempH = real*8 variable - temporary height for calculating decay rate
*    upalt = real*8 variable - altitude at which to reboost to
*    upprop = real*8 variable - amount of prop available for reboost
*    waste = real*8 variable - total amount of propellant waste per lifetime
*    wasteprop = real*8 variable - amount of propellant bleeded off after reboost
*    world = integer counter variable for the number of three-cycle worlds to simulate
*    year = integer variable to determine time
************************************************************************
************************************************************************
*
*              DECAY/REBOOST  MODEL MAIN PROGRAM
*
************************************************************************
      Program decmod1
      implicit none
```

```
*////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR DECAY/REBOOST MODEL
      integer pworlds
      parameter (pworlds=10)
      real*8 sflux(pworlds,0:195), solar_flux
      real*8 pflux(pworlds,0:195)
      real*8 density, endprop, rebalt,wasteprop,waste,onprop
      real*8 progM2, minreb
      real*8 H, newH, p, mass, BN,BC,B,Bshutt
      real*8 upalt,T3alt
      real*8 height(pworlds,5500), info(pworlds,0:16)
      real*8 fuel(pworlds,5500),penalty(pworlds,75)
      real*8 Re,mu,Cd,dayseconds,minalt,upprop
      logical fake,help
      integer month,day,dday, boost, world,m, year,i,d
      integer mgdays,wrlds, slip,k
      integer calendar(pworlds,5500),event(pworlds,5500)
      integer microg(pworlds,15), restricted(pworlds,5500)
      integer sshut,shuttle(pworlds,75)
      real*8 r, drand
      integer seed
*---INITIALIZE VARIABLES FOR DECAY/REBOOST MODEL
      Re = 6378.135 ! (km) earth radius
      mu = 3.98601e5 !(km)^3/(sec)^2 gravitational parameter
      Cd = 2.3 !ballistic coefficient
      mass = 426376 !(kg) mass of station
      progM2 = 2300 !(kg) Amount of useable propellant in Progress M2
      rebalt = 300          !(km) Default altitude for reboost
      upalt = 0
      dayseconds = 24.*60.*60. !seconds in a day
      BN = 14               !mass/(Cd*A) in lb/ft^2
      BN = BN*4.882427636       !Convert to kg/m^2
      BC = 1.e-6/BN             !km^2/kg, Cd*(A)/mass (ballistic coefficient)
      Bshutt = B/.9             !ballistic coefficient ISS w/ shuttle attached
      boost = 0             !Counter for number of reboosts accomplished

      wrlds = pworlds
      do day = 1,5400
       do world = 1,wrlds
         calendar(world,day) = 0
         event(world,day) = 0
         restricted(world,day) = 0
       enddo
      enddo
*////////////////////////////////////////////////
*/ OPEN FILES FOR DECAY/REBOOST MODEL
*////////////////////////////////////////////////
*--- Input files
     open(unit=7,file='supercyc.in',status='old')
     open(unit=14,file='predcyc.in',status='old')
*--- Output files
* List of world, year, and day of reboost
```

```
      open(unit=8,file='decayreb.dat',status='unknown')
* List of all the worlds daily altitude
      open(unit=9,file='height.dat',status='unknown')
* List of total days of microgravity per year for each world
      open(unit=10,file='microg.dat',status='unknown')
* List of total number of reboosts per year for each world
      open(unit=11,file='reboost.dat',status='unknown')
* List of a randomly chosen world, the altitude and the events
* associated with it
      open(unit=24,file='dci65.dat',status='unknown')
      open(unit=25,file='dci61.dat',status='unknown')
      open(unit=28,file='dci22.dat',status='unknown')
      open(unit=30,file='dci26.dat',status='unknown')
      open(unit=32,file='dci46.dat',status='unknown')
* List of the daily capacity of the storage tanks of the ISS
      open(unit=13,file='prop.dat',status='unknown')
* List of the penalties for the shuttle
      open(unit=15,file='shuttle.dat',status='unknown')
*////////////////////////////////////////////////
*/ INITIALIZE RANDOM UNIFORM(0,1) GENERATOR
*////////////////////////////////////////////////
* Initialize random number generator and then call uniform(0,1) random
* variable r=drand(0)
      seed = 7651234
      r = drand(seed)
*---READ IN ALTITUDE STRATEGY
*              call strategy(altstrat,rebalt,useprop,H)
*////////////////////////////////////////////////
*/ READ IN HISTORICAL SOLAR DATA
*////////////////////////////////////////////////
      if (mu*(Re+H).gt.0) then    ! Makes sure altitude is greater than 0
       do world=1,wrlds   ! Number of different 3-cycle worlds
        do month = 1,195
         read(7,*) sflux(world,month)   ! Read "actual" solar activity
*  Convert sunspot number to solar flux values using NASA TM-82478 equation
         sflux(world,month)=49.4+(0.97*sflux(world,month))+17.6*
     &                 (exp(-.035*sflux(world,month)))
         read(14,*) pflux(world,month)  ! Read "predicted" solar activity
         pflux(world,month)=49.4+(0.97*pflux(world,month))+17.6*
     &                 (exp(-.035*pflux(world,month)))
        enddo
       enddo
       write(8,110)
110    format(2x,'WORLD',5x,'YEAR',5x,'DAY')
*/////////////////////////////////////////////////////////////////////
*/ SCHEDULE SHUTTLE AND SOYUZ FLIGHTS - currently the same for all worlds
*/ & INITIALIZE MICRO-GRAVITY COUNTS
*/////////////////////////////////////////////////////////////////////
       do world = 1,wrlds
        sshut = 0
        do year = 1,15
         microg(world,year) = 0      !Initialize days of Micro-gravity
*--- SCHEDULE SHUTTLE (Time on station = 7 days)
         do i = 0,4
```

```
      sshut = sshut + 1
      r = drand(0)
      slip = int(r*10)
      day = (year-1)*360 + 65 + slip + i*72
      shuttle(world,sshut) = day
      event(world,day) = 1       ! Shuttle docks, microG disturbed
      calendar(world,day) = 1
      do d=day,day+6
       restricted(world,d) = 1   ! Designates shuttle is attached
      enddo
      day = day + 6
      event(world,day) = 2       ! Shuttle undocks, microG disturbed
     enddo                  ! End Shuttle scheduling loop
*--- SCHEDULE SOYUZ (Time on station is about 180 days)
     do i = 0,1
* According to March 21,1995 TM Report, a Soyuz is docked for about 6 months
* and a second one is launched nine days earlier than the first one departs.
* We are assuming that it takes 2 days between launch and dock.
      day = (year-1)*360 + 30 + i*171
      d = day
55     if(restricted(world,d).eq.0) then ! Make sure Soyuz doen't preempt other events
       calendar(world,d) = 2
       event(world,d) = 1          ! Soyuz docks, microG disturbed
      else
       d = d-1
       goto 55
      endif
      day = d+180
      d = day
56     if(restricted(world,d).eq.0) then ! Make sure Soyuz can undock (Shuttle not on)
       calendar(world,d) = 3
       event(world,day) = 1        ! Soyuz undocks, microG disturbed
      else
       d = d-1
       goto 56
      endif
     enddo  ! End Soyuz scheduling loop
    enddo   ! End year loop
   enddo    ! End world loop
*////////////////////////////////////////////////
*/ MEAT OF THE PROGRAM
*////////////////////////////////////////////////
*---START WORLD LOOP
   do world=1,wrlds
    write(6,*) world
* (Re)Initialize variables at the start of each new world
    onprop = 3200      !(kg)  Default propellant onboard at start of world
    fuel(world,1) = onprop
    B = BC             ! Initialize Ballistic Coefficient without shuttle
    sshut = 0          ! Initialize the first shuttle to 0
    shutt = 1          ! Initialize the first shuttle to 1
    boost = 0          ! Initialize number of reboosts to 0
    k = 1
    mgdays = 0         ! Initialize number of days of microgravity to 0
```

```
      minreb = 60           ! Allows a reboost to occur any time at first
      waste = 0            ! Initialize waste propellant to zero
      year=1              ! Initialize year
      month = 1             ! Initialize month
      dday = 1             ! Initialize actual day (ranges from 1-5400)
      tempday = 0
* Initialize random height
      call TTT(pflux,world,dday,onprop,B,minalt,T3alt)
       H = T3alt
*---START MONTH LOOP
      do month  = 1,180
       if(mod(month,20).eq.0) then
        WRITE(*,*) ('Month'),month
       endif
       solar_flux = sflux(world,month)  ! Assume constant solar flux for! 30 days
*---CALCULATE DENSITY
       p = density(H,solar_flux)
*---START DAY LOOP
       do day=1,30
        dday = (month-1)*30 + day    ! Current day in 15 year simulation (1 to 5400)
* Check and see if the shuttle is attached
        if (restricted(world,day).eq.1) then
         B = Bshutt
        else
         B = BC
        endif
* Check and see if the shuttle is docking
        if (calendar(world,dday).eq.1) then
         sshut = sshut+1
         penalty(world,sshut) = H-358
        endif
        if(mod(dday,10).eq.0) then
         call TTT(pflux,world,dday,onprop,B,minalt,T3alt)
        endif
*---CALCULATE ALTITUDE
        H = H - sqrt(mu*(Re+H))*B*p*dayseconds
        if (H.gt.0) then         ! Make sure we are above ground
*---UPDATE DENSITY
         p = density(H,solar_flux)    ! Calculate density
        else
         write (*,*) ('WARNING, WE CRASHED!')
         write (*,*) world,month,day
         goto  189
         stop
        endif
        if (H.lt.T3alt.and.minreb.ge.60) then
****REBOOST
         fake = .true.
         upprop = 3200
         call reboost(H,H,upprop,newH,onprop,wasteprop,
   &              fake,help)
         waste = waste + wasteprop
         H = newH
         boost = boost+1
```

```
          minreb = 0
         else
          ! OK, keep going
          minreb = minreb+1
        if(world.eq.22.or.world.eq.26.or.world.eq.46
     &      .or.world.eq.61.or.world.eq.65) then
          fuel(world,dday) = endprop
        endif
         endif
*/////////////////////////////////////////////////
*/ BEGIN RECORDING CALCULATIONS
*/////////////////////////////////////////////////
*---RECORD HEIGHT
       if(world.eq.22.or.world.eq.26.or.world.eq.46
     &      .or.world.eq.61.or.world.eq.65) then
         height(world,k) = H
       endif
         k=k+1
*---CALCULATE DAYS OF MICROGRAVITY
         if (event(world,dday).eq.0) then    ! If no disturbances, then the number of
           mgdays = mgdays + 1               ! microgravity days increases by one
         else ! Disturbance occurs
           if (mgdays.ge.30) then    ! Only blocks of 30 or more days of microgravity are
             microg(world,year) = microg(world,year)+mgdays    ! included in the total amount
           endif                                     ! of microG per year
           mgdays = 0   ! Reset count of microG back to zero
         endif
       enddo  ! End day do-loop
*---CALCULATE NUMBER OF REBOOST PER YEAR
       if(mod(month,12).eq.0) then
         info(world,year) = boost   !Number of reboosts for the year
         year = year + 1
         boost = 0
       endif
      enddo    ! End month do-loop
*---CALCULATE WASTED PROPELLANT PER YEAR
      info(world,16) = waste          ! wasted propellant for world lifetime
     enddo     ! End world do-loop
    else
     write(6,25)
25    format(2x,'WARNING - Our altitude is not above zero!')
    endif     ! End altitude if-loop
*/////////////////////////////////////////////////
*/ BEGIN WRITING INFORMATION TO FILES
*/////////////////////////////////////////////////
* Write "Days of microgravity per year" to file
* Write "Initial Altitude" "Reboosts/year" and "Wasted Prop" to file
189  write(11,675)
675  format(1x,'Initial Altitude',2x,'Reboosts/year',2x,
     &      65x,'Wasted Prop')
    do world = 1,wrlds
      write(10,725) (microg(world,i),i=1,15)
725     format(2x,15(i5,3x))
      write (11,750) (info(world,m), m=0,16)
```

```
750    format (2x,f10.5,5x,15(3x,f3),5x,f10.1)
   enddo
* Write events on the calendar to a file
   do day = 1,5400
     world = 65
     write(24,775)world,day,height(world,day),
    &           fuel(world,day)
     world = 61
     write(25,775)world,day,height(world,day),
    &           fuel(world,day)
     world = 22
     write(28,775)world,day,height(world,day),
    &           fuel(world,day)
     world = 26
     write(30,775)world,day,height(world,day),
    &           fuel(world,day)
     world = 46
     write(32,775)world,day,height(world,day),
    &           fuel(world,day)
775    format (i5,3x,i5,3x,f5,5x,f7,5x,i2)
   enddo
* Write shuttle penalties to a file (shuttle.dat)
   write(15,800) world
800  format(2x,10(i6,8x))
   do sshut=1,75
     write(15,810) (penalty(i,sshut),i=1,wrlds)
810    format(2x,10(f10.5,4x))
   enddo
   write(6,*) 'end!'
*//////////////////////////////////////////////////
*/ CLOSE FILES
*//////////////////////////////////////////////////
   close(7)
   close(8)
   close(9)
   close(10)
   close(11)
   close(24)
   close(25)
   close(28)
   close(30)
   close(32)
999  end
***************************************************************************
* FUNCTION  DENSITY    for Decay/Reboost Model
*    Global variables
*      alt = real*8 variable - function reads in the ISS's height above earth's surface
*      density = real*8 variable - function outputs the density of the atmosphere
*      sflux = real*8 variable - contains current solar_flux (solar activity)
*    Local varaibles
*      a = real*8 variable - density exponent at 150 km (10^(-a) = density at 150 km)
*      b = real*8 variable - density exponent at 500 km (10^(-b) = density at 500 km)
*      hialt = real*8 variable - upper allowable altitude (in km)
*      LC = real*8 variable used in calculation of density
```

```
*     loalt = real*8 variable - lower allowable altitude (in km)
*     x0 = real*8 variable used in claculation of density
*     Y = exponent of the density
**************************************************************************
    FUNCTION density(alt,solar_flux)
    implicit none
*///////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*///////////////////////////////////////////////
*--- DECLARE VARIABLES FOR DENSITY FUNCTION
    real*8 density    ! output global variable
    real*8 alt,solar_flux  ! input global variables
    real*8 loalt,hialt,a,b,x0,LC,Y   !local variables
*--- INITIALIZE VARIABLES FOR DENSITY FUNCTION
    loalt = 150    ! lower allowable altitude
    hialt = 500    ! upper altitude (460 is max allowable)
    a = 8.73       ! density for all curves at 150 km (in kg/m^3)
    b = 13-((1.6*(solar_flux-70))/(245-70))
              ! density at 500 km (varies 11.4 to 13 for 245 to 70 hi/lo)
*///////////////////////////////////////////////
*/ MEAT OF FUNCTION
*///////////////////////////////////////////////
    x0 = ((log(hialt)*a)-(log(loalt)*b))/(log(hialt)-log(loalt))
    LC = (log(loalt))/(a-x0)
    Y = (log(alt)/LC) + x0
    density = (10**(-Y))*(1.e9)
    return
    end

**************************************************************************
* SUBROUTINE  REBOOST   for Decay/Reboost Model
*    Global variables
*     upht = real*8 variable - the altitude we are trying to reboost ISS to (km)
*     loht = real*8 variable - the altitude that ISS is at before reboost (km)
*     newht = real*8 variable - altitude ISS is at after reboost (km)
*     stprop = real*8 variable is the propellant desired (available) for reboost (kg)
*     endprop = real*8 variable equals (available propellant - used propellant)
*     waste = real*8 variable that contains any propellant not used that must be thrown
*          overboard because storage tanks are full
*    fake = logical variable to determine if this is a real reboost or not
*    help = logical variable to determine if we are out of propellant for a reboost
*    Local variables
*     atx
*     ddeltaV = real*8 variable for desired velocity in reboost in (kg)
*     deltaV = real*8 variable for actual velocity in reboost in (kg)
*     diff = real*8 variable for difference between usable propellant and actual
*          propellant used in (kg)
*     G    = constant variable - Earth's gravitational acceleration (km/s^2)
*     i = integer counter variable
*     Isp  = constant variable - Specific Impulse of Propellant (sec)
*     mass = constant variable - mass of ISS
*     mu   = constant variable - gravitational parameter = Ge*M in (km)^3/(sec)^2
*     prop2 = real*8 variable - actual amount of propellant used in reboost in (kg)
*     rA   = real*8 variable - Lower altitude for reboost equals (loht+Rearth) in (km)
```

```
*     rB  = real*8 variable - Altitude reboosted to equals (newht + Rearth) in (km)
*     Rearth = constant variable - radius of the earth (km)
*     stopper = integer 0-1 variable
*********************************************************************
      SUBROUTINE reboost(loht,upht,stprop,newht,endprop,waste,fake,help)
       implicit none
*///////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*///////////////////////////////////////////////
*--- DECLARE VARIABLES FOR REBOOST SUBROUTINE
       integer pworlds
       parameter (pworlds=100)
       real*8 loht,upht,stprop,newht,endprop,waste    ! Global variables
       logical fake,help                              ! Global variables
       real*8  deltaV, ddeltaV                        ! Local variables
       real*8 rA, rB, atx,diff, prop2                 ! Local variables
       real*8 Rearth,mu,Isp,G,mass                    ! Local variables
       integer i,stopper                              ! Local variables
*--- INITIALIZE VARIABLES FOR REBOOST SUBROUTINE
       Rearth = 6378.135 ! Radius of the earth
       mass = 419119.3  ! Mass of ISSA (kg)
       Isp  = 230     ! Specific Impulse of Propellant (sec)
       G    = .0098    ! Earth's gravitational acceleration (km/s^2)
       mu = 3.98601e5  ! Gravitational parameter (km)^3/(sec)^2
       rA  = loht+Rearth      ! Lower altitude for reboost (km)
       ddeltaV = -G*Isp*dlog(1-(stprop/mass))
       rB = rA
       deltaV = 0
       diff = abs(ddeltaV*1000-deltaV*1000)
       i=0
*///////////////////////////////////////////////
*/ MEAT OF SUBROUTINE
*///////////////////////////////////////////////
      if(fake.eq..false.) then
       if (loht.eq.upht) then
        newht = loht
        endprop = stprop+endprop
       else
        rB = upht+Rearth
        if (rB.lt.rA) then
         write(6,24)
24        format('You are trying to reboost downward, you idiot')
        endif
        atx = (rA+rB)/2
        deltaV=sqrt(mu)*((abs((sqrt((2/rA)-(1/atx)))-(sqrt(1/rA))))
     &         +(abs((sqrt((2/rB)-(1/atx)))-(sqrt(1/rB)))))
        prop2=mass*(1-exp(-deltaV/(G*Isp)))
        if ((stprop-prop2).lt.0.0) then
         help = .true.
         if (stprop.gt.2300) then
          write(*,*) ('HELP, WE ARE REALLY OUT OF PROP!')
          stop
         endif
         goto 999
```

```
        stop
      else
        help = .false.
        stopper = 0
        endprop = stprop-prop2
        newht = rB-Rearth
        if (endprop.gt.5927) then   ! Max capacity of FGB+SM
          waste = endprop-5927      ! Bleeded off propellant
          endprop = 5927
        endif
      endif
    endif
  elseif (fake.eq..true.) then
    stopper=1
    do while (diff.gt.0.5.and.stopper.eq.1)
      if ((rB-Rearth).ge.460) then
        stopper = 0
      else
        i = i+1
        rB = rB+.1
      endif
      atx = (rA + rB)/2
      deltaV=sqrt(mu)*((abs((sqrt((2/rA)-(1/atx)))-(sqrt(1/rA))))
 &          +(abs((sqrt((2/rB)-(1/atx)))-(sqrt(1/rB)))))
      diff = abs(ddeltaV*1000-deltaV*1000)
      newht = rB-Rearth
    enddo
  else
    write(*,*) ('Error in Reboost Subroutine')
  endif
999 return
  end

*************************************************************************
* SUBROUTINE  TTT   for Decay/Reboost Model
*   Global variables
*     pflux = real*8 array - predicted solar activity values
*     world = integer - world number we are currently at
*     dday = integer - current day in simulation (1-5400)
*     onprop = real*8 - amount of propellant we can use (kg)
*     B = real*8 - ballistic coefficient
*     minalt = real*8 - minimum altitude we can decay to
*     T3alt = real*8 - altitude where T3 occurs
*   Local variables
*     t = integer - equals day of T3 simulation
*     first = integer - first day of T3 simulation
*     last = integer - last day of T3 simulation
*     mnth = month of T3 simulation
*     rem = integer to determine month
*     pf = real*8 - predicted density

  SUBROUTINE TTT(pflux,world,dday,onprop,B,minalt,T3alt)
  implicit none
*///////////////////////////////////////////////
```

```
*/ VARIABLE INTRODUCTION
*///////////////////////////////////////////////
*--- DECLARE VARIABLES FOR DECAY SUBROUTINE
     integer pworlds
     parameter (pworlds=100)
     real*8 pflux(pworlds,0:195)          ! Global variables
     real*8 minalt,T3alt,onprop,B          ! Global variables
     integer world,dday                    ! Global variables
     real*8 decalt,pf,upprop,ht,h,newH      ! Local variables
     logical fake,help                     ! Global variables
     integer i,t,first,last,rem,mnth  ! Local variables
     real*8 Re,mu,dayseconds,endprop      ! Local variables
     real*8 waste,wasteprop,p,density   ! Local variables

     Re = 6378.135 ! Radius of the earth
     mu = 3.98601e5   ! Gravitational parameter (km)^3/(sec)^2
     dayseconds = 24.*60.*60. !seconds in a day
     first = dday
     T3alt = 0
     last = first+360
     H = 300.
      do while (T3alt.eq.0)
       upprop = onprop
       H = H + 1
       fake = .true.
       call reboost(H,H,upprop,newH,onprop,wasteprop,
    &            fake,help)
       decalt = newH
       t = dday-1
       mnth = int(dday/30) + 1
       do i = first,last
        t = t+1
        if (mod(i,30).eq.0) then
         mnth = int(i/30) + 1
         pf = pflux(world,mnth)
        endif
        p = density(decalt,pf)
        decalt = decalt-sqrt(mu*(Re+decalt))*B*p*dayseconds
        if (decalt.lt.278) then
          T3alt = 0
          goto 36
        endif
       enddo
       T3alt = H
36     enddo
999   return
     end

************************************************************************
* SUBROUTINE predecay   for Decay/Reboost Model
*   Global variables (not listed/defined variables have been used before)
*     alt = real*8 - current altitude
*     predalt = real*8 - altitude needed for T3
*   Local variables
```

```
************************************************************************
      SUBROUTINE predecay(alt,pflux,world,dday,B,sshut,predalt)
      implicit none
*////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR DECAY SUBROUTINE
      integer pworlds
      parameter (pworlds=100)
      real*8 alt,pflux(pworlds,0:195),B    ! Global variables
      real*8 predalt                       ! Global variables
      integer world,dday,sshut             ! Global variables
      real*8 Re,mu,dayseconds              ! Local variables
      real*8 pf,p,density                  ! Local variables
      integer first,last,t,rem,i,mnth      ! Local variables

      Re = 6378.135 ! Radius of the earth
      mu = 3.98601e5  ! Gravitational parameter (km)^3/(sec)^2
      dayseconds = 24.*60.*60. !seconds in a day
      first = dday
      last = dday+sshut
      mnth = int(dday/30)+1
      predalt = alt

      do i = first,last
       t = t+1
       rem = mod(i-1,30)        ! Check to see if we are at a new month
       if(rem.eq.0) then
         mnth = ((i-1)/30)+1
         pf = pflux(world,mnth)
       endif
       p = density(predalt,pf)
       predalt = predalt-sqrt(mu*(Re+predalt))*B*p*dayseconds
       if (predalt.lt.278) then
         predalt = 0
         goto 36
       endif
      enddo

36    return
      end
```

Appendix E: Decay/Reboost FORTRAN Program for Phase IV

```
*************************************************************************
*      TITLE: Decay/Reboost Model Fortran Program (dmIV.for) Phase III
*     AUTHOR: 2d Lt Jillene B. Rylaarsdam and Major E. Price Smith
*       DATE: March 1996
* DESCRIPTION: This model reads in the different three-cycle worlds from a file and
*          stores the monthly solar data in an array. The data is read in as the
*          12-month smoothed Zurich sunspot number and then converted to the solar
*          flux value using NASA's equation from TM-82478. A loop allows many
*          three-cycle worlds to be run. The number of reboosts completed in a world
*          begins at zero, as does the "wasted" propellant. The minimum number of
*          days between a reboost is 60. The initial altitude begins at an altitude
*          that is 360 days to 278 km. The density at the altitude is calculated
*          daily and the T3 altitude is calculated every 5 days. The T3
*          altitude is defined as the altitude that the station is at after reboosting
*          and decaying for 360 days. If the altitude after 360 days of decay
*          is less than or equal to 278 km (360 days to 278 km is required by NASA)
*          and the 60 day between reboost constraint is not violated, the station will
*          be reboosted. A Progress M2 will be used for the reboost. The reboost
*          will be accomplished in an attempt to use all available onboard propellant as
*          well as Progress propellant in order to get the station down to T3 8 days after
*          the shuttle docks (7 days shuttle is attached, 8th day reboost occurs). This is
*          to try to bring the station down lower for the shuttle so that the shuttle can have
*          more upmass. The reboost first attempts to reboost the station to T3. If this is
*          not possible due to lack of fuel, it reboosts it as high as it can (as long as it is
*          less than 460 km). An iteration scheme was used to determine the amount
*          of propellant to use, with a given tolerance between the desired and the actual.
*          If the station is reboosted to 460 km (Russian hardware design constraint)
*          the reboost will terminate and any remaining propellant will first transfer to
*          the station and, once full, the remaining waste propellant will be thrown
*          overboard. Waste propellant will be calculated through the lifetime. The
*          daily altitude, waste propellant, and number of boosts will be kept as
*          statistics.
*
*  FILES USED:
*  Input file: supercyc.in     "Actual" many three-cycle worlds to simulate
*              predcyc.in      Predicted many three-cycle worlds to simulate
* Output files: decayreb.dat   Keeps track of month,day,height,and density
*              height.dat      Keeps track of height and outputs in a format to be
*                              used by Excel
*              prop.dat        Keeps track of propellant data
*              decinfo.dat     Keeps track of initial altitude, number of reboosts,
*                              and waste per three-cycle world lifetime
*              dci__.dat Keeps track of altitude and fuel level for Excel
*              microg.dat      Calculates the number of days of micro-G per year
*                              (in blocks of 30 or more)
*              reboost.dat     Calculates the number of reboosts per year
*              shuttle.dat     Keeps track of shuttle penalties per shuttle
* Hierarchy of the program, subroutines and functions
*
*                    DECMOD
```

```
*                  / | \ \
*           predecay   TTT | |
*                  /   \| |
*                  | Reboost |
*                  \  |  /
*                   density
*   FUNCTIONS:
*         density      calculates the density given the altitude and level
*                      of solar activity
*  SUBROUTINES:
*         predecay     determines the altitude the ISS would end up at if
*                      no reboosts occurred for 360 days.  If the altitude is
*                      less than 278 km, then it determines the altitude that we
*                      need to reboost to in order to get 360 days to 278 km.
*         reboost      given the altitude and amount of propellant available
*                      for use, reboosts the ISS using the available propellant
*         TTT          determines the altitude that the station needs to be at in
*                      order for the station to be at about 278 km in 360 days after
*                      reboosting the station with all available onboard propellant.
* VARIABLES:
* Main Program:
*   B = constant for the ballistic coefficient quantity = Cd*A/mass
*       Cd = constant for the coefficient of drag
*       A = presented area of the space station
*       m = mass of the space station
*   BC = ballistic coefficient w/out shuttle attached
*   BN = constant for the ballistic number = mass/(Cd*A)
*   boost = integer counter variable to caculate number of boost in 180 months
*   Bshutt = Ballistic coefficient w/ shuttle attached
*   calendar(world,day) = integer time array to get a calendar input
*       Events:  1 = Shuttle dock/undock  2 = Soyuz dock/undock
*                3 = Soyuz undock       4 = Progress undock
*   case =
*   d = integer counter variable for a loop
*   day  = integer counter variable for the number of days to simulate (30 per month)
*   dd = integer variable annotating if the Progress can undock (no shuttle attached)
*   dday = integer variable annotating the day the reboost occurs (1-5400)
*   dayseconds = real*8 variable to convert days to seconds
*   decay = real*8 variable annotating the altitude (in km) that the ISS would be at if
*        no reboosts occurred
*   density = real*8 function variable (see above description)
*   drand = real*8 intrinsic function variable - outputs random, uniform(0,1) varaite
*   endprop = real*8 variable - stores the amount of propellant left after reboost
*   event(world,day) = indicates microgravity disturbances, used to calculate quiet periods
*   fake = logical variable to determine if we are doing a fake reboost or not
*        .true. = fake reboost   .false. = real reboost
*   fuel(world,day) = real*8 variable designating the current amount of propellant
*              in the storage tanks of the ISS
*   H = real*8 variable - height of ISS above the surface of the earth
*   height(world,day) = real*8 array - stores the heights of the ISS during its lifetime
*         for each of the three-cycle worlds calculated
*   help = logical variable to determine if we are out of propellant for a reboost
*   i = integer counter variable
*   i100th = real*8 variable that retrieves the 100th of a second from the time
```

```
*     ihr = real*8 variable that retrieves the hour from the time
*     imin = real*8 variable that retrieves the minute from the time
*     info(world,0:16) = real*8 array that keeps track of vital information per three-cycle
*           0 = initial altitude
*           1 thru 15 = number of reboosts years 1-15
*           16 = amount of wasted propellant per world lifetime
*     isec = real*8 variable that retrieves the second from the time
*     iseed = real*8 function variable - combines the time variables to find a random seed
*     m = integer counter variable
*     mgdays = integer counter for days of microgravity
*     mass = constant - mass of the ISS
*     microg(world,year) = integer array that keeps track of days of micro-gravity each year
*     minreb = integer counter - number of days since last reboost, updated daily (to make sure
*           that there are at least 20 days between reboosts)
*     month = integer counter variable for the number of months to simulate per cycle
*     mu = constant - gravitational parameter = Ge*M in (km)^3/(sec)^2
*         Ge = Universal gravitational constant = 6.67x10^(-11) in Nm^2/kg^2
*         M  = mass of the earth  (kg)
*     newalt = real*8 variable
*     newH = real*8 variable - new height of ISS after reboost
*     onprop - real*8 variable - amount of propellant onboard (kg)
*     p = real*8 variable - equals 'density' variable after calculating function
*     penalty(worlds,75) = real*8 array - penalty for docking other than 358 km
*           '+' = bad (got a penalty)  '-' = good (negative penalty)
*     pflux(world,0:195) = real*8 array reads in the "predicted" sunspot data and then is
*           converted to solar flux data.
*     progM2 = constant - available useable propellant from Progress M2  (kg)
*     pworlds = parameter - the number of runs we are simulating
*     r = real random uniform(0,1) variable
*     Re = constant - radius of the earth (km)
*     rebalt = real*8 variable - lower altitude from which to reboost in (km)
*     restricted(world,day) = integer array that does not allow either Progress or Soyuz
*                 to dock or undock while the shuttle is attached
*     seed = real*8 predetermined seed set by programmer used for random number generation
*     shutt = integer - day of next shuttle
*     sshut = integer value for the shuttle number = 1-75  (5 per world)
*     shuttle = 0-1 variable to determine if shuttle is attached
*           0 = shuttle is not attached   1 = shuttle is attached
*     sflux(world,0:195) = real*8 array reads in the "actual" sunspot data and then is
*           converted to solar flux data.
*     skipper = integer variable to determine if the days between reboosts is at least 60
*     slip = real*8 variable - equals +(0-10) days that the shuttle takes off
*     solar_flux = real*8 variable equal to sflux(world,month) for each loop
*     sshut = integer variable - index for the day the shuttle undocks
*     sT3 = real*8 - T3 altitude at the shuttle rendezvous
*     T3alt = altitude of station to make 278 km in 360 days w/ reboost using all prop
*     tempH = real*8 variable - temporary height for calculating decay rate
*     upalt = real*8 variable - altitude at which to reboost to
*     upprop = real*8 variable - amount of prop available for reboost
*     waste = real*8 variable - total amount of propellant waste per lifetime
*     wasteprop = real*8 variable - amount of propellant bleeded off after reboost
*     world = integer counter variable for the number of three-cycle worlds to simulate
*     year = integer variable to determine time
**********************************************************************
```

```
************************************************************************
*
*                 DECAY/REBOOST  MODEL MAIN PROGRAM
*
************************************************************************
      Program decmod2
      implicit none
*/////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*/////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR DECAY/REBOOST MODEL
      integer pworlds
      parameter (pworlds=100)
      real*8 sflux(pworlds,0:195), solar_flux
      real*8 pflux(pworlds,0:195)
      real*8 density, endprop, rebalt,wasteprop,waste,onprop
      real*8 progM2, minreb
      real*8 H, newH, p, mass, BN,BC,B,Bshutt
      real*8 upalt,T3alt
      real*8 height(pworlds,5500), info(pworlds,0:16)
      real*8 fuel(pworlds,5500),penalty(pworlds,75)
      real*8 Re,mu,Cd,dayseconds,minalt,upprop,sT3
      logical fake,help
      integer month,day,dday, boost, world,m,year,i,d,k
      integer mgdays,wrlds,slip
      integer calendar(pworlds,5500),event(pworlds,5500)
      integer microg(pworlds,15), restricted(pworlds,5500)
      integer sshut,shuttle(pworlds,75),shutt
      real*8 r, drand
      integer seed
*---INITIALIZE VARIABLES FOR DECAY/REBOOST MODEL
      Re = 6378.135 ! (km) earth radius
      mu = 3.98601e5 !(km)^3/(sec)^2 gravitational parameter
      Cd = 2.3 !ballistic coefficient
      mass = 426376 !(kg) mass of station
      progM2 = 2300 !(kg) Amount of useable propellant in Progress M2
      rebalt = 300          !(km) Default altitude for reboost
      upalt = 0
      dayseconds = 24.*60.*60. !seconds in a day
      BN = 14               !mass/(Cd*A) in lb/ft^2
      BN = BN*4.882427636       !Convert to kg/m^2
      BC = 1.e-6/BN            !km^2/kg, Cd*(A)/mass (ballistic coefficient)
      Bshutt = B/.9          !ballistic coefficient ISS w/ shuttle attached
      boost = 0            !Counter for number of reboosts accomplished
      wrlds = pworlds
      do day = 1,5400
       do world = 1,wrlds
        calendar(world,day) = 0
        event(world,day) = 0
        restricted(world,day) = 0
       enddo
      enddo
*/////////////////////////////////////////////////
*/ OPEN FILES FOR DECAY/REBOOST MODEL
```

```
*///////////////////////////////////////////////
*--- Input files
     open(unit=7,file='supercyc.in',status='old')
     open(unit=14,file='predcyc.in',status='old')
*--- Output files
* List of world, year, and day of reboost
     open(unit=8,file='decayreb.dat',status='unknown')
* List of all the worlds daily altitude
     open(unit=9,file='height.dat',status='unknown')
* List of total days of microgravity per year for each world
     open(unit=10,file='microg.dat',status='unknown')
* List of total number of reboosts per year for each world
     open(unit=11,file='reboost.dat',status='unknown')
* List of a randomly chosen world, the altitude and the events
* associated with it
     open(unit=12,file='decinfo.dat',status='unknown')
     open(unit=24,file='dci65.dat',status='unknown')
     open(unit=25,file='dci61.dat',status='unknown')
     open(unit=28,file='dci22.dat',status='unknown')
     open(unit=30,file='dci26.dat',status='unknown')
     open(unit=32,file='dci46.dat',status='unknown')
* List of the daily capacity of the storage tanks of the ISS
     open(unit=13,file='prop.dat',status='unknown')
* List of the penalties for the shuttle
     open(unit=15,file='shuttle.dat',status='unknown')
*///////////////////////////////////////////////
*/ INITIALIZE RANDOM UNIFORM(0,1) GENERATOR
*///////////////////////////////////////////////
* Initialize random number generator and then call uniform(0,1) random
* variable r=drand(0)
     seed = 7651234
     r = drand(seed)
*///////////////////////////////////////////////
*/  READ IN HISTORICAL SOLAR DATA
*///////////////////////////////////////////////
     if (mu*(Re+H).gt.0) then    ! Makes sure altitude is greater than 0
      do world=1,wrlds   ! Number of different 3-cycle worlds
       do month = 1,195
        read(7,*) sflux(world,month)   ! Read "actual" solar activity
*  Convert sunspot number to solar flux values using NASA TM-82478 equation
        sflux(world,month)=49.4+(0.97*sflux(world,month))+17.6*
    &                  (exp(-.035*sflux(world,month)))
        read(14,*) pflux(world,month)  ! Read "predicted" solar activity
        pflux(world,month)=49.4+(0.97*pflux(world,month))+17.6*
    &                  (exp(-.035*pflux(world,month)))
       enddo
      enddo
      write(8,110)
110   format(2x,'WORLD',5x,'YEAR',5x,'DAY')
*//////////////////////////////////////////////////////////////////
*/  SCHEDULE SHUTTLE AND SOYUZ FLIGHTS - currently the same for all worlds
*/  & INITIALIZE MICRO-GRAVITY COUNTS
*//////////////////////////////////////////////////////////////////
      do world = 1,wrlds
```

```
        sshut = 0
        do year = 1,15
          microg(world,year) = 0      !Initialize days of Micro-gravity
*--- SCHEDULE SHUTTLE (Time on station = 7 days)
          do i = 0,4
            sshut = sshut + 1
            r = drand(0)
            slip = int(r*10)
            day = (year-1)*360 + 65 + slip + i*72
            shuttle(world,sshut) = day
            event(world,day) = 1         ! Shuttle docks, microG disturbed
            calendar(world,day) = 1
            do d=day,day+6
              restricted(world,d) = 1    ! Designates shuttle is attached
            enddo
            day = day + 6
            event(world,day) = 2         ! Shuttle undocks, microG disturbed
          enddo                    ! End Shuttle scheduling loop
*--- SCHEDULE SOYUZ (Time on station is about 180 days)
          do i = 0,1
* According to March 21,1995 TM Report, a Soyuz is docked for about 6 months
* and a second one is launched nine days earlier than the first one departs.
* We are assuming that it takes 2 days between launch and dock.
            day = (year-1)*360 + 30 + i*171
            d = day
55            if(restricted(world,d).eq.0) then ! Make sure Soyuz doesn't preempt other events
              calendar(world,d) = 2
              event(world,d) = 1           ! Soyuz docks, microG disturbed
            else
              d = d-1
              goto 55
            endif
            day = d+180
            d = day
56            if(restricted(world,d).eq.0) then ! Make sure Soyuz can undock (Shuttle not on)
              calendar(world,d) = 3
              event(world,day) = 1           ! Soyuz undocks, microG disturbed
            else
              d = d-1
              goto 56
            endif
          enddo  ! End Soyuz scheduling loop
        enddo    ! End year loop
      enddo      ! End world loop
*///////////////////////////////////////////////
*/  MEAT OF THE PROGRAM
*///////////////////////////////////////////////
*---START WORLD LOOP
      do world=1,wrlds
       write(6,*) world
* (Re)Initialize variables at the start of each new world
       minalt = 278
       onprop = 3200       !(kg)  Default propellant onboard at start of world
       fuel(world,1) = onprop
```

```
      B = BC              ! Initialize Ballistic Coefficient without shuttle
      sshut = 1           ! Initialize the first shuttle to 0
      shutt = shuttle(world,1)! Initialize day of the first shuttle
      boost = 0           ! Initialize number of reboosts to 0
      k = 1
      mgdays = 0            ! Initialize number of days of microgravity to 0
      minreb = 60           ! Allows a reboost to occur any time at first
      waste = 0           ! Initialize waste propellant to zero
      year=1             ! Initialize year
      month = 1            ! Initialize month
      dday = 1            ! Initialize actual day (ranges from 1-5400)
* Initialize random height
       call TTT(pflux,world,dday,onprop,B,minalt,T3alt)
       H = T3alt
       call TTT(pflux,world,shutt,onprop,B,minalt,T3alt)
       sT3 = T3alt
       call decay (world,dday,shutt,H,sT3,pflux,B,newH)
       H = newH
*---START MONTH LOOP
      do month  = 1,180
        if (mod(month,20).eq.0) then
          WRITE(*,*) ('Month'),month
        endif
        solar_flux = sflux(world,month)  ! Assume constant solar flux for! 30 days
*---CALCULATE DENSITY
        p = density(H,solar_flux)
*---START DAY LOOP
        do day=1,30
         dday = (month-1)*30 + day    ! Current day in 15 year simulation (1 to 5400)
* Check and see if the shuttle is attached
         if (restricted(world,day).eq.1) then
           B = Bshutt
         else
           B = BC
         endif
* Check and see if the shuttle is docking
         if (calendar(world,dday).eq.1) then
           penalty(world,sshut) = H-358
           sshut = sshut+1
           if (sshut.le.75) then
             shutt = shuttle(world,sshut)
           else
             shutt = 5350
           endif
         endif
         if(mod(dday,10).eq.0) then
           call TTT(pflux,world,dday,onprop,B,minalt,T3alt)
         endif
*************************
*---CALCULATE ALTITUDE
*************************
         H = H - sqrt(mu*(Re+H))*B*p*dayseconds
         if (H.gt.0) then          ! Make sure we are above ground
*---UPDATE DENSITY
```

```
          p = density(H,solar_flux)    ! Calculate density
         else
          write (*,*) ('WARNING, WE CRASHED!')
          write (*,*) world,month,day
          goto 189
          stop
         endif
         if (H.lt.T3alt.and.minreb.ge.60) then
****REBOOST
          call TTT(pflux,world,shutt,onprop,B,minalt,T3alt)
          sT3 = T3alt
          upprop = progM2+onprop
          fake=.false.
          call decay (world,dday,shutt,H,sT3,pflux,B,newH)
          call reboost(H,newH,upprop,newH,onprop,wasteprop,
     &                 fake,help)
          waste = waste + wasteprop
          H = newH
          boost = boost+1
          minreb = 0
         else
          ! OK, keep going
          minreb = minreb+1
          fuel(world,dday) = endprop
         endif
*/////////////////////////////////////////////////
*/ BEGIN RECORDING CALCULATIONS
*/////////////////////////////////////////////////
*---RECORD HEIGHT
         height(world,k) = H
         k=k+1
*---CALCULATE DAYS OF MICROGRAVITY
         if (event(world,dday).eq.0) then    ! If no disturbances, then the number of
           mgdays = mgdays + 1              ! microgravity days increases by one
         else ! Disturbance occurs
           if (mgdays.ge.30) then     ! Only blocks of 30 or more days of microgravity are
            microg(world,year) = microg(world,year)+mgdays    ! included in the total amt
           endif                                              ! of microG per year
           mgdays = 0   ! Reset count of microG back to zero
         endif
        enddo  ! End day do-loop
*---CALCULATE NUMBER OF REBOOST PER YEAR
        if(mod(month,12).eq.0) then
         info(world,year) = boost   !Number of reboosts for the year
         year = year + 1
         boost = 0
        endif
       enddo    ! End month do-loop
*---CALCULATE WASTED PROPELLANT PER YEAR
      info(world,16) = waste          ! wasted propellant for world lifetime
     enddo    ! End world do-loop
    else
     write(6,25)
25    format(2x,'WARNING - Our altitude is not above zero!')
```

```
      endif     ! End altitude if-loop
*/////////////////////////////////////////////////
*/ BEGIN WRITING INFORMATION TO FILES
*/////////////////////////////////////////////////
* Write "Days of microgravity per year" to file
* Write "Initial Altitude" "Reboosts/year" and "Wasted Prop" to file
189  write(11,675)
675  format(1x,'Initial Altitude',2x,'Reboosts/year',2x,
   &      65x,'Wasted Prop')
      do world = 1,wrlds
        write(10,725) (microg(world,i),i=1,15)
725     format(2x,15(i5,3x))
        write (11,750) (info(world,m), m=0,16)
750     format (2x,f10.5,5x,15(3x,f3),5x,f10.1)
      enddo
* Write events on the calendar to a file
      do day = 1,5400
        world = 65
        write(24,775)world,day,height(world,day),
   &             fuel(world,day)
        world = 61
        write(25,775)world,day,height(world,day),
   &             fuel(world,day)
        world = 22
        write(28,775)world,day,height(world,day),
   &             fuel(world,day)
        world = 26
        write(30,775)world,day,height(world,day),
   &             fuel(world,day)
        world = 46
        write(32,775)world,day,height(world,day),
   &             fuel(world,day)
775     format (i5,3x,i5,3x,f5,5x,f7,5x,i2)
      enddo
* Write shuttle penalties to a file (shuttle.dat)
      write(15,800) world
800  format(2x,10(i6,8x))
      do sshut=1,75
        write(15,810) (penalty(i,sshut),i=1,wrlds)
810     format(2x,10(f10.5,4x))
      enddo
      write(6,*) 'end!'
*/////////////////////////////////////////////////
*/ CLOSE FILES
*/////////////////////////////////////////////////
      close(7)
      close(8)
      close(9)
      close(10)
      close(11)
      close(24)
      close(25)
      close(28)
      close(30)
```

```
      close(32)
999   end
**************************************************************************
* FUNCTION DENSITY    for Decay/Reboost Model
*     Global variables
*       alt = real*8 variable - function reads in the ISS's height above earth's surface
*       density = real*8 variable - function outputs the density of the atmosphere
*       sflux = real*8 variable - contains current solar_flux (solar activity)
*     Local varaibles
*       a = real*8 variable - density exponent at 150 km (10^(-a) = density at 150 km)
*       b = real*8 variable - density exponent at 500 km (10^(-b) = density at 500 km)
*       hialt = real*8 variable - upper allowable altitude (in km)
*       LC = real*8 variable used in calculation of density
*       loalt = real*8 variable - lower allowable altitude (in km)
*       x0 = real*8 variable used in claculation of density
*       Y = exponent of the density
**************************************************************************
      FUNCTION density(alt,solar_flux)
      implicit none
*//////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*//////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR DENSITY FUNCTION
      real*8 density     ! output global variable
      real*8 alt,solar_flux  ! input global variables
      real*8 loalt,hialt,a,b,x0,LC,Y   !local variables
*--- INITIALIZE VARIABLES FOR DENSITY FUNCTION
      loalt = 150     ! lower allowable altitude
      hialt = 500     ! upper altitude (460 is max allowable)
      a = 8.73        ! density for all curves at 150 km (in kg/m^3)
      b = 13-((1.6*(solar_flux-70))/(245-70))
               ! density at 500 km (varies 11.4 to 13 for 245 to 70 hi/lo)
*//////////////////////////////////////////////////
*/ MEAT OF FUNCTION
*//////////////////////////////////////////////////
      x0 = ((log(hialt)*a)-(log(loalt)*b))/(log(hialt)-log(loalt))
      LC = (log(loalt))/(a-x0)
      Y = (log(alt)/LC) + x0
      density = (10**(-Y))*(1.e9)
      return
      end
**************************************************************************
* SUBROUTINE REBOOST   for Decay/Reboost Model
*     Global variables
*       upht = real*8 variable - the altitude we are trying to reboost ISS to (km)
*       loht = real*8 variable - the altitude that ISS is at before reboost (km)
*       newht = real*8 variable - altitude ISS is at after reboost (km)
*       stprop = real*8 variable is the propellant desired (available) for reboost (kg)
*       endprop = real*8 variable equals (available propellant - used propellant)
*       waste = real*8 variable that contains any propellant not used that must be thrown
*           overboard because storage tanks are full
*      fake = logical variable to determine if this is a real reboost or not
*      help = logical variable to determine if we are out of propellant for a reboost
*     Local variables
```

```
*     atx
*     ddeltaV = real*8 variable for desired velocity in reboost in (kg)
*     deltaV = real*8 variable for actual velocity in reboost in (kg)
*     diff = real*8 variable for difference between usable propellant and actual
*          propellant used in (kg)
*     G    = constant variable - Earth's gravitational acceleration (km/s^2)
*     i = integer counter variable
*     Isp  = constant variable - Specific Impulse of Propellant (sec)
*     mass = constant variable - mass of ISS
*     mu   = constant variable - gravitational parameter = Ge*M in (km)^3/(sec)^2
*     prop2 = real*8 variable - actual amount of propellant used in reboost in (kg)
*     rA   = real*8 variable - Lower altitude for reboost equals (loht+Rearth) in (km)
*     rB   = real*8 variable - Altitude reboosted to equals (newht + Rearth) in (km)
*     Rearth = constant variable - radius of the earth (km)
*     stopper = integer 0-1 variable
***********************************************************************
      SUBROUTINE reboost(loht,upht,stprop,newht,endprop,waste,fake,help)
       implicit none
*//////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*//////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR REBOOST SUBROUTINE
       integer pworlds
       parameter (pworlds=100)
       real*8 loht,upht,stprop,newht,endprop,waste     ! Global variables
       logical fake,help                               ! Global variables
       real*8  deltaV, ddeltaV                         ! Local variables
       real*8 rA, rB, atx,diff, prop2                  ! Local variables
       real*8 Rearth,mu,Isp,G,mass                     ! Local variables
       integer i,stopper                               ! Local variables
*--- INITIALIZE VARIABLES FOR REBOOST SUBROUTINE
       Rearth = 6378.135 ! Radius of the earth
       mass = 419119.3  ! Mass of ISSA (kg)
       Isp  = 230       ! Specific Impulse of Propellant (sec)
       G    = .0098     ! Earth's gravitational acceleration (km/s^2)
       mu = 3.98601e5   ! Gravitational parameter (km)^3/(sec)^2
       rA   = loht+Rearth      ! Lower altitude for reboost (km)
       ddeltaV = -G*Isp*dlog(1-(stprop/mass))
       rB = rA
       deltaV = 0
       diff = abs(ddeltaV*1000-deltaV*1000)
       i=0
*//////////////////////////////////////////////////
*/ MEAT OF SUBROUTINE
*//////////////////////////////////////////////////
      if(fake.eq..false.) then
       if (loht.eq.upht) then
        newht = loht
        endprop = stprop+endprop
       else
        rB = upht+Rearth
        if (rB.lt.rA) then
         write(6,24)
24        format('You are trying to reboost downward, you idiot')
```

```
        stop
      endif
      atx = (rA+rB)/2
      deltaV=sqrt(mu)*((abs((sqrt((2/rA)-(1/atx)))-(sqrt(1/rA))))
   &         +(abs((sqrt((2/rB)-(1/atx)))-(sqrt(1/rB)))))
      prop2=mass*(1-exp(-deltaV/(G*Isp)))
      if ((stprop-prop2).lt.0.0) then
        help = .true.
        if (stprop.gt.2300) then
          fake=.true.
          rB=rA
          goto 326
          write(*,*) ('HELP, WE ARE REALLY OUT OF PROP!')
          stop
        endif
        goto 999
        stop
      else
        help = .false.
        stopper = 0
        endprop = stprop-prop2
        newht = rB-Rearth
        if (endprop.gt.5927) then   ! Max capacity of FGB+SM
          waste = endprop-5927      ! Bleeded off propellant
          endprop = 5927
        endif
      endif
    endif
   elseif (fake.eq..true.) then
326    stopper=1
    do while (diff.gt.0.5.and.stopper.eq.1)
      if ((rB-Rearth).ge.460) then
        stopper = 0
      else
        i = i+1
        rB = rB+.1
      endif
      atx = (rA + rB)/2
      deltaV=sqrt(mu)*((abs((sqrt((2/rA)-(1/atx)))-(sqrt(1/rA))))
   &         +(abs((sqrt((2/rB)-(1/atx)))-(sqrt(1/rB)))))
      diff = abs(ddeltaV*1000-deltaV*1000)
      newht = rB-Rearth
    enddo
   else
    write(*,*) ('Error in Reboost Subroutine')
   endif
999  return
   end
************************************************************************
* SUBROUTINE  TTT   for Decay/Reboost Model
*    Global variables
*       pflux = real*8 array - predicted solar activity values
*       world = integer - world number we are currently at
*       dday = integer - current day in simulation (1-5400)
```

```
*       onprop = real*8 - amount of propellant we can use (kg)
*       B = real*8 - ballistic coefficient
*       minalt = real*8 - minimum altitude we can decay to
*       T3alt = real*8 - altitude where T3 occurs
*     Local variables
*       t = integer - equals day of T3 simulation
*       first = integer - first day of T3 simulation
*       last = integer - last day of T3 simulation
*       mnth = month of T3 simulation
*       rem = integer to determine month
*       pf = real*8 - predicted density
*************************************************************
      SUBROUTINE TTT(pflux,world,dday,onprop,B,minalt,T3alt)
      implicit none
*//////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*//////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR TTT SUBROUTINE
      integer pworlds
      parameter (pworlds=100)
      real*8 pflux(pworlds,0:195)          ! Global variables
      real*8 minalt,T3alt,onprop,B         ! Global variables
      integer world,dday                   ! Global variables
      real*8 decalt,pf                     ! Local variables
      real*8 upprop,h,newH
      logical fake,help
      integer i,t,first,last,mnth          ! Local variables
      real*8 Re,mu,dayseconds              ! Local variables
      real*8 wasteprop,p,density           ! Local variables

      Re = 6378.135 ! Radius of the earth
      mu = 3.98601e5  ! Gravitational parameter (km)^3/(sec)^2
      dayseconds = 24.*60.*60. !seconds in a day
      first = dday
      T3alt = 0
      last = first+360
      H = 300.
       do while (T3alt.eq.0)
        upprop = onprop
        H = H + 1
        fake = .true.
        call reboost(H,H,upprop,newH,onprop,wasteprop,
     &            fake,help)
        decalt = newH
        t = dday-1
        mnth = int(dday/30) + 1
        do i = first,last
         t = t+1
         if (mod(i,30).eq.0) then
          mnth = int(i/30) + 1
          pf = pflux(world,mnth)
         endif
         p = density(decalt,pf)
         decalt = decalt-sqrt(mu*(Re+decalt))*B*p*dayseconds
```

```
      if (decalt.lt.minalt) then
        T3alt = 0
        goto 36
      endif
     enddo
     T3alt = H
36   enddo
999  return
     end

*************************************************************************
* SUBROUTINE  decay   for Decay/Reboost Model
*    Global variables   (Variables not listed or defined have been used before)
*    Local variables


     SUBROUTINE decay(world,dday,shutt,H,sT3,pflux,B,newH)
     implicit none
*/////////////////////////////////////////////////
*/ VARIABLE INTRODUCTION
*/////////////////////////////////////////////////
*--- DECLARE VARIABLES FOR DECAY SUBROUTINE
     integer pworlds
     parameter (pworlds=100)
     real*8 H,pflux(pworlds,0:195),B     ! Global variables
     real*8 sT3,newH                     ! Global variables
     integer world,dday,shutt            ! Global variables
     real*8 Re,mu,dayseconds             ! Local variables
     real*8 pf,p,density                 ! Local variables
     real*8 upalt,alt,Dalt
     integer first,last,t,rem,i,mnth     ! Local variables

     Re = 6378.135 ! Radius of the earth
     mu = 3.98601e5   ! Gravitational parameter (km)^3/(sec)^2
     dayseconds = 24.*60.*60. !seconds in a day
     first = dday
     last =  shutt+8
     if((last-first).lt.60) then
       last = first+60
     endif
     mnth = int(dday/30)+1
     newH = H
     upalt = H
     Dalt = 0
     alt = H

     do while (Dalt.lt.sT3)
      upalt = upalt+1
      if (upalt.ge.460) then
       goto 777
      endif
      alt = upalt
      do i = first,last
       t = t+1
```

```
          rem = mod(i-1,30)         ! Check to see if we are at a new month
          if(rem.eq.0) then
            mnth = ((i-1)/30)+1
            pf = pflux(world,mnth)
          endif
          p = density(alt,pf)
          alt = alt-sqrt(mu*(Re+alt))*B*p*dayseconds
          if (alt.lt.sT3) then
            Dalt = 0
            goto 36
          endif
        enddo
        Dalt = upalt
36      enddo
777   newH = upalt
      return
      end
```

Bibliography

Banke, Jim. "The View From Balkonur," *Ad Astra* **7(3)**: 23 (July 1995).

Boden, Daryl G. "Introduction to Astrodynamics." *Space Mission Analysis and Design* (Second Edition). Eds. Wiley J. Larson and James R. Wertz. Torrance, CA: Microcosm, Inc. and Dordrecht: Kluwer Academic Publishers, 1992. 129-156

Boeing Missiles & Space Division, Defense & Space Group, NASA. *International Space Station Alpha: Reference Guide*. Incremental Design Review (IDR) #1. March 29, 1995. http://ISSA-www.jsc.nasa.gov/ss/techdata/ISSAR/ISSAReferenceGuide.html

Cowen, Ron. "Space: The International Approach," *Science News* **147(20):** 312-314 (May 20, 1995).

Cowen, Ron. "Space Station: Merger with the Russians," *Science News* **144:** 399 (December 11, 1993).

Delaney, Russell.A. McDonnell Douglas Space Systems - Houston Division. Transmittal Memo # 5.25.01-124, 13 November 1992.

Delaney, Russell A., McDonnell Douglas Space Systems - Houston Division. *STRAP User Handbook Input Delivery.* Transmittal Memo # 94.150-3, March 21, 1994.

Eastman, Ford. "Kennedy Orders Defense Review, Boost in Airlift, Missile Programs." *Aviation Week & Space Technology* **74(6)**: 46 (February 6, 1961).

Easterbrook, Gregg. "The Case Against NASA." *New Republic* : 17-24 (July 8, 1991).

Gorney, D.J.. "Solar Cycle Effects on the Near-Earth Space Environment." *Reviews of Geophysics* **28(3)**: 315-336 (August 1990).

Hartman, James K. *Lecture Notes In High Resolution Combat Modeling 1985.* Class Notes, OPER 671, Joint Combat Modeling I. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, Summer Quarter 1995.

Hax, Arnold C. and Dan Candea. *Production and Inventory Management.* London: Prentice-Hall International, Inc., 1984.

Hedin, A.E. "MSIS-86 Thermospheric Model." *Space Mission Analysis and Design,* (Second Edition). Eds. Wiley J. Larson and James R. Wertz. Torrance, CA: Microcosm, Inc. and Dordrecht: Kluwer Academic Publishers, 1992.

Hillier, Frederick S. and Gerald J. Lieberman. *Introduction to Operations Research* (Fifth Edition). New York: McGraw-Hill Publishing Company, 1990.

Koepke, Dean, Computer Sciences Corporation, and Brian McDonald, McDonnell Douglas Space Systems Company - Houston Division. *STRAP v.5.0 Programmer's Guide*. Transmittal Memo # CSC2-BB-286. November 19, 1992.

Kolcum, Edward H. "Delta Engine Shutdown Investigation Centers on Electrical Relay Box." *Aviation Week & Space Technology* **124(19)**: 20-22 (May 12, 1986).

Larson, Wiley J. and James R. Wertz , eds. *Space Mission Analysis and Design* (Second Edition). Torrance, CA: Microcosm, Inc. and Dordrecht: Kluwer Academic Publishers, 1992.

Law, Averill M. and W. David Kelton. *Simulation Modeling and Analysis* (Second Edition). New York: McGraw-Hill, Inc., 1991.

Lemmons, Neil, Rockwell Space Operations Company for National Aeronautics and Space Administration (NASA). *International Space Station Traffic Model: Version 2.0 Requirements*. Lyndon B. Johnson Space Center: Space Station Program Office, Houston, Texas. Preliminary 26 June 1995.

Loyd, Arnold J. *TDS 3.1.1-3; ISSA Altitude Profile Assessment*. February 28, 1995.

McDonald, Brian and Robert Puckett, McDonnell Douglas Space Systems Company - Houston Division, Houston, Texas. *Altitude Strategy and Propellant Utilization for Space Station Freedom.* McDonnell Douglas Space Systems Company - Houston Division. November 22, 1991.

McDonald, Brian M. and Scott B. Teplitz, McDonnell Douglas Space Systems Company - Houston Division, Houston, Texas. *Space Station Freedom Altitude Strategy.* AIAA Space Programs & Technologies Converence & Exhibit. Huntsville, Alabama. September 25-27, 1990.

McKenna, James T.. "Shuttle Makes Historic Mir Docking." *Aviation Week & Space Technology* **143(1)**: 18-19 (July 3, 1995).

Mykytka, Edward F. and Paul F. Auclair. Class Notes, OPER 660, Simulation Modeling and Analysis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, Fall Quarter 1995.

National Aeronautics and Space Administration (NASA). *Jacchia-Lineberry Upper Atmosphere Density Model.* Lyndon B. Johnson Space Center: Houston, Texas. October 1982.

National Aeronautics and Space Administration (NASA). "Skylab & Apollo-Soyuz." 2 September 1995. http://www.msfc.nasa.gov/online/msfc/spacelink3.html

National Aeronautics and Space Administration (NASA). "Space Station White Papers." 2 September 1995. http://issa-www.jsc.nasa.gov/ss/prgview/sswp.html

Pinedo, Michael. *Scheduling: Theory, Algorithms, and Systems.* London: Prentice-Hall, 1995.

Pritsker, A. Alan B. *Introduction to Simulation and SLAM II.* New York: Halsted Press Book, John Wiley & Sons, 1986.

Project Skylab. 2 September 1995. http//www.ksc.nasa.gov/history/skylab/skylab.html

Puckett, Robert J., McDonnell Douglas Aerospace - Houston Division. *DAC-1: Propellant Resource Assessment TDS 3.1.1-4. Final Report.* Transmittal Memo #95-0034-04. March 30, 1995.

Puckett, Robert J., McDonnell Douglas Space Systems - Houston Division. "ISSA Altitude Strategy DDP - Vait Presentation." Transmittal Memo # 95-0034-01 16 December 1994.

Sanders, Roger and Robert Puckett, McDonnell Douglas Aerospace - Houston Division. *ISSA Altitude Strategy DDP - VAIT Presentation* Transmittal Memo #95-0034-01. December 16, 1994.

Schwabe, H., "Solar Observations During 1843," *Astronomische Nacrichten* **21:** 223-236, (December 1843), translated into English by editor A.J. Meadows, *Early Solar Physics*, London: Pergamon Press, 1970.

Sellers, Jerry J. and others. *Understanding Space: An Introduction to Astronautics.* New York: McGraw-Hill, Inc., 1994.

Shishko, Robert, Ph.D., National Aeronautics and Space Administration (NASA). *NASA Systems Engineering Handbook.* SP-6105. June 1995.

Solomon, Marius. "Algorithms for the Vehicle-Routing and Scheduling Problems with Time Window Constraints", *Operations Research,* 35 (2): 254-265 (1987)

Smith, Rober E. and George S. West, comp., National Aeronautics and Space Administration (NASA). *Space and Planetary Environment Criteria Guidelines for Use in Space Vehicle Development, 1982 Revision (Volume 1)*, NASA Technical Memorandum 82478, January 1983.

Tascione, Thomas F. *Introduction to the Space Environment*. Florida: Orbit Book Company, 1988.

Thompson, Richard. IPS Radio & Space Services, Sydney, Australia. http://www.ips.oz.au/papers/richard/ssn.def.html. November 7, 1995.

"U.S. Launcher Crisis." *Aviation Week & Space Technology* **124(12)**: 11 (April 28, 1986).

Walterscheid, R.L. "The Upper Atmosphere." *Space Mission Analysis and Design,* (Second Edition). Eds. Wiley J. Larson and James R. Wertz. Torrance, CA: Microcosm, Inc. and Dordrecht: Kluwer Academic Publishers, 1992. 206-212.

Wiesel, William. Professor, AFIT WPAFB, Ohio. Personal Interview. 3 November 3, 1995.

Wiesel, William E. *Space Flight Dynamics*. New York: McGraw-Hill Publishing Company, 1989.

Winston, Wayne L. *Operations Research: Applications and Algorithms* (Third Edition). California: Duxbury Press, 1994.

Vita

Lieutenant Jillene B. Rylaarsdam [redacted] She graduated from Unity Christian High School, in Hudsonville, Michigan, in June of 1990 and attended the United States Air Force Academy in Colorado Springs, Colorado. After graduating with a Bachelor of Science in Operations Research in June of 1994, she was assigned to the Graduate Operations Analysis program in the School of Engineering at the Air Force Institute of Technology, Wright-Patterson AFB, Ohio. Following her March 1996 graduation, Lieutenant Rylaarsdam received an assignment as a space systems analyst at Los Angeles AFB, El Segundo, California.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1996	Master's Thesis

4. TITLE AND SUBTITLE

International Space Station Traffic Modeling and Simulation

5. FUNDING NUMBERS

6. AUTHOR(S)

Jillene B. Rylaarsdam, 2Lt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
WPAFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOA/ENS/96M-08

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Johnson Space Center
NASA/OC
Attn: Jessica Kite
NASA Road 1
Houston, TX 77058

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In an effort to provide NASA with an alternative perspective and some insights to the operational planning of the International Space Station (ISS), this research developed a simulation environment for the ISS and devised a method to evaluate various altitude strategies. The simulation environment allowed us to incorporate the natural random behaviors which affect the lifetime of objects in low-earth orbit. We created prototype models of the operational planning process to analyze current altitude strategy approaches and acquire new strategies from insights observed. In addition, by extrapolating random future solar activity values from the interpolation of historical data, we established a spectrum of possible solar activity rather than just maximum, mean, and minimum values. From this process, we demonstrated a procedure to analyze a strategy using distributions of parameter outputs in response to random inputs.

14. SUBJECT TERMS

Simulation, Modeling, Solar Activity, Altitude Profile, Altitude Strategy
Random inputs

15. NUMBER OF PAGES

152

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL